1-1-2008

Evaluating Mobility Predictors in Wireless Networks for Improving Handoff and Opportunistic Routing

Libo Song
Dartmouth College

Evaluating Mobility Predictors in Wireless Networks for Improving Handoff and Opportunistic Routing

Dartmouth Technical Report TR2008-611

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

in

Computer Science

by

Libo Song

DARTMOUTH COLLEGE

Hanover, New Hampshire

January 2008

Examining Committee:

(chair) David F. Kotz, Ph.D.

Andrew T. Campbell, Ph.D.

Thomas H. Cormen, Ph.D.

Ravi Jain, Ph.D.

Charles K. Barlowe, Ph.D.
Dean of Graduate Studies

Abstract

We evaluate mobility predictors in wireless networks. Handoff prediction in wireless networks has long been considered as a mechanism to improve the quality of service provided to mobile wireless users. Most prior studies, however, were based on theoretical analysis, simulation with synthetic mobility models, or small wireless network traces. We study the effect of mobility prediction for a large realistic wireless situation.

We tackle the problem by using traces collected from a large production wireless network to evaluate several major families of handoff-location prediction techniques, a set of handoff-time predictors, and a predictor that jointly predicts handoff location and time. We also propose a fallback mechanism, which uses a lower-order predictor whenever a higher-order predictor fails to predict.

We found that low-order Markov predictors, with our proposed fallback mechanisms, performed as well or better than the more complex and more space-consuming compression-based handoff-location predictors. Although our handoff-time predictor had modest prediction accuracy, in the context of mobile voice applications we found that bandwidth reservation strategies can benefit from the combined location and time handoff predictor, significantly reducing the call-drop rate without significantly increasing the call-block rate.

We also developed a prediction-based routing protocol for mobile opportunistic networks. We evaluated and compared our protocol's performance to five existing routing protocols, using simulations driven by real mobility traces. We found that the basic routing protocols are not practical for large-scale opportunistic networks. Prediction-based routing protocols trade off the message delivery ratio against resource usage and performed well and comparable to each other.

Acknowledgments

First, I thank my adviser, David Kotz. David has helped me clarify, justify and arrange my ideas throughout this project; he has been, and I am sure will continue to be, an excellent mentor.

I thank all the examining committee members: David Kotz, Andrew Campbell, Tom Cormen, and Ravi Jain, for their time to read my thesis and comments that have improved it.

I thank my wife, Wei Li, for her great support when I was struggling and frustrated with my research.

I thank my parents and my parents-in-law for their love and support.

I thank all the members of the Center of Mobile Computing at Dartmouth College, and especially Tristan Henderson and Minkyong Kim for their advice, discussion and encouragement. I thank my fellow Ph.D student, Udayan Deshpande, for his hard work on the simulation of the case study of this project.

I also thank our collaborators from DoCoMo USA Labs, Ravi Jain, Ulas Kozat, and Xiaoning He for many thought-provoking meetings, and constructive comments and suggestions.

I thank Judith Hertog for helping improve my English and writing.

I thank the staff of Computer Science and Computing Services at Dartmouth College for their assistance in collecting the data.

I thank the Weber family: Bryan, Mary Jo, and Zachary. We lived together for about two years. They helped on almost everything other than academic issues.

This research is a project of the Center for Mobile Computing and the Institute for Security Technology Studies at Dartmouth College. It was supported by DoCoMo Labs USA, the CRAWDAD archive at Dartmouth College (funded by NSF CRI Award 0454062), NSF Infrastructure Award EIA-9802068, and by Grant number 2005-DD-BX-1091 awarded by the Bureau of Justice Assistance. Points of view or opinions in this document are those of the author and do not represent the official position or policies of any sponsor.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Wireless-network technology frees people from fixed locations and penetrates into many aspects of people's lives. Cellular networks are mature for voice communications around the world. Wireless local area networks (WLANs) are increasingly popular: "Wi-Fi" is used to access the Internet and enterprise networks, and Bluetooth technology enables device communication in a personal area network. A fundamental problem in mobile computing and wireless networks is the ability to track and predict the location of mobile devices as they roam, or "handoff," from one access point (AP) to another. An accurate handoff location and time predictor can significantly improve the performance or reliability of wireless network protocols, the wireless network infrastructure itself, and many applications in pervasive computing. These improvements lead to a better user experience, to a more cost-effective infrastructure, or both. In mobile ad hoc networks, especially those networks with sparse population and frequent link breaks, estimating the probability with which users meet each other can improve the routing protocols. First, we introduce handoff prediction in wireless infrastructure networks.

1.1 Prediction in wireless infrastructure networks

The flexibility of mobility introduces challenges in guaranteeing quality of service (QoS) for voice, video, and other real-time traffic. QoS is affected by the packet transmission rate, packet delay, and network connectivity. When a wireless user associated with one access point moves to another AP, it disassociates from the previous AP and starts association with the new AP. This procedure is called "handoff." If the handoff procedure is slow, packets will be lost or delayed. Anticipating the next AP can allow the destination AP to prepare for the new client and reduce the handoff delay. Thus, we can improve the QoS.

Some applications, such as voice or video communications, require a certain minimum bandwidth. If the required bandwidth cannot be fulfilled by the network, the connection has to be dropped. In the literature, it is believed that forced termination of an ongoing call is more objectionable than blocking a new call request [KS03]. In cellular networks, bandwidth can be reserved for incoming handoff calls. WLAN standards, including Wi-Fi or Bluetooth networks, do not provide bandwidth-reservation mechanisms. We envision a future in which WLANs will provide some sort of bandwidth-reservation mechanisms for real-time traffic. Before a wireless device roams to the next AP, predicting the destination and time of the handoff may help the network to reserve sufficient bandwidth for the user at the next AP, reducing the probability of dropped calls, which improves the QoS.

Wireless networks also support pervasive computing. Many pervasive computing applications include opportunities for location-based services and information. Anticipating a user's *next* location can improve the user experience with some pervasive applications. With a location predictor, these applications can provide services or information based on the user's next location. For example, consider a university student who moves from dormitory to classroom to dining hall to library. When the student snaps a photo of his classmates using a wireless digital camera and wishes to print it, the camera's printing

application might suggest printers near the current location and near the next predicted location, enabling the photo to finish printing while the student walks across campus.

The above cases benefit from predicting the next location. Handoff time or location prediction is the process of predicting the time or location (destination AP) of the user's next handoff. Here, we emphasize two points: (i) prediction accuracy largely depends on the granularity of the available data set and (ii) the success of a predictor strongly depends on how close its assumptions about the user mobility model the real situation. In this dissertation, we use the terms *location*, *cell*, and *access point (AP)* interchangeably. *Position* should be understood as a point in space.

Every predictor is limited by the granularity of the available information. There are three common options.

1. Record the visited locations and time of handoff to these locations for each user. Since this information is readily available to every network operator that supports roaming, it is widely used [LAN97, LB96, YL02, CS98, LBC98].

2. Record the position and velocity of each user. This information can be obtained by signal triangulation techniques or client-carried GPS devices, both increasingly common in cellular phones. Here, mobility prediction requires algorithms that can infer cell boundaries and user velocity [LBC98, SMY00, AZ01, SK01, SN02].

3. Use high-level information that describes the distinguishing, designating or limiting properties of each cell, derived from road and building maps [LB96, SK03, SK04].

Location prediction has been proposed in many areas of wireless cellular networks as a means of enhancing performance, including better mobility management [LM96], improved assignment of cells to location areas [DS99], more efficient paging [BD02, KT04], call admission control [YL02], and location-based services [KL03]. Most of them, however, were evaluated by simulations using synthetic data, or small traces of user mobility.

A fundamental question is how these prior proposed predictors would fare with *real* user mobility patterns, in large populations and over long time periods. Because obtaining real users' mobility data is difficult, researchers have had trouble evaluating predictors with real data. Although cellular networks are mature, it is almost impossible for researchers to get confidential user mobility data from the main wireless service providers.

Fortunately, WLANs are becoming more and more popular at universities, enterprise campuses, and metropolitan areas. It is relatively easy for researchers to get data from WLAN administrators. At Dartmouth College, we have collected "syslog" messages from our campus-wide wireless network for several years. From these messages, which contain a record of association and re-association events, we extracted traces of users' location and handoff events from one location to another. We evaluated prediction methods using our real wireless mobility data. We evaluated four families of predictors, namely Markov, LZ, PPM, and SPM predictors. We found that low-order Markov predictors performed as well or better than the more complex and more space-consuming compression-based predictors.

In addition to predicting the next AP, it is important to anticipate a user's handoff *time* to support applications such as bandwidth reservation, which needs to know when to reserve bandwidth. It is impossible to predict the exact time of the next handoff. What we can do is to predict the most likely time period during which the handoff will occur, or to estimate the probability of handoff within a period of time. We developed Markov, Moving Average, and cumulative distribution function (CDF) time predictors that can predict the next handoff time in precision of minutes or hours. We also can combine a location predictor and a time predictor to compute the probability that a user hands off to a certain AP within a given period of time.

We must evaluate time predictions differently from location predictions. In location predictions, the predictor predicts the next AP, and we define the prediction as correct when our prediction matches the next AP. In time predictions, however, the predictor predicts a

period of time during which a handoff will occur, or it predicts the probability of handoff within a given time period. To evaluate the performance of the time predictors, we design other metrics. One new metric is to measure the lateness or earliness of the predicted time from the actual handoff time. We do not measure one the magnitude of the error of the predicted time, because some applications may be more sensitive to early predictions and others may be more sensitive to late predictions.

To evaluate the combined location-and-time predictor, we defined new metrics for "over-provisioning" and "under-provisioning." These metrics were specifically designed for the bandwidth-reservation application, which reserved bandwidth at APs according to the user's probability to visit them next. We also used the call-drop rate and call-block rate to evaluate our predictor performance when simulating several bandwidth-reservation schemes using our location-and-time integrated predictor on real mobility data. The results show that we can reduce call-drop rate significantly without a significant deterioration of the call-block rate.

Since our simulation indicates that with accurate location-and-time prediction the QoS of calls can be improved, we believe studying and better understanding the characteristics of wireless users can improve the performance of predictors. From the results of our location or time predictions, we found that we could predict accurately for some users, and poorly for other users. That means that some users had mobility characteristics to be more predictable than others. We then studied the predictability of users by correlating their prediction accuracy with the entropy of their movements. It is not surprising that these two were closely correlated, because the entropy reflects the uncertainty of the user traces. Low entropy means low uncertainty, thus high prediction accuracy.

1.2 Prediction in opportunistic networks

The above scenarios assume that mobile devices communicate through a wireless-network infrastructure. In many cases, no infrastructure exists; mobile wireless devices may form ad hoc networks and communicate with each other. When the population density is low, a contemporaneous end-to-end path may not exist for every pair of users. One user may only communicate with another when they are in a connected subnetwork. We call these networks *mobile opportunistic networks*.

Mobile opportunistic networks are one kind of delay-tolerant network (DTN) [Fal03]. Delay-tolerant networks provide service despite long link delays or frequent link breaks. Long link delays happen in networks with communication between nodes at a great distance, such as interplanetary networks [BHT+03]. Link breaks are caused by nodes moving out of range, environmental changes, interference from other moving objects, radio power-offs, or failed nodes. For us, mobile opportunistic networks are those DTNs with sparse node population and frequent link breaks caused by power-offs and the mobility of the nodes.

Mobile opportunistic networks have received increasing interest from researchers. In the literature, these networks include mobile sensor networks [WW06], wild-animal tracking networks [JOW+02], "pocket-switched" networks [HCS+05], and transportation networks [BGJL06, LCGZ05]. We expect to see more opportunistic networks when the one-laptop-per-child (OLPC) project [OLP] starts rolling out inexpensive laptops with wireless networking capability for children in developing countries, where often no infrastructure exists. Opportunistic networking is one promising approach that can enable those children to exchange information.

One fundamental problem in opportunistic networks is how to route messages from their source to their destination. Mobile opportunistic networks differ from the Internet

in that disconnections are the norm instead of the exception. In mobile opportunistic networks, communication devices can be carried by people [CEL+06], vehicles [BGJL06] or animals [JOW+02]. Some devices can form a small mobile ad hoc network when the nodes move close to each other, but a node may frequently be isolated from other nodes. Note that traditional Internet routing protocols and mobile ad hoc network (MANET) routing protocols, such as AODV [PR99] or DSDV [PB94], assume that a contemporaneous end-to-end path exists, and thus fail in mobile opportunistic networks. Indeed, there may never exist an end-to-end path between two given devices. Store-carry-forward approaches are popular routing schemes in opportunistic networks. Predicting node contacts can help nodes to make good routing decisions.

We study protocols for routing messages between wireless networking devices carried by people. We assume that people send messages to other people occasionally, using their devices; when no direct link exists between the source and the destination of the message, other nodes may relay the message to the destination. Each device represents a unique person (we do not consider in this thesis situations in which a single device may be carried by multiple people). Each message is destined for a specific person and thus for a specific node carried by that person. Although one person may carry multiple devices, we assume that the sender knows which device is the best to receive the message. We do not consider multicast or geocast in this dissertation.

Many routing protocols have been proposed in the literature. Few of them were evaluated in realistic network settings, or even in realistic simulations, due to the lack of any realistic people-mobility model. Random walk or random way-point mobility models are often used to evaluate the performance of those routing protocols. Although these synthetic mobility models have received extensive interest by mobile ad hoc network researchers [CBD02], they do not reflect people's mobility patterns [JLB05]. Realising the limitations of using random mobility models in simulations, a few researchers have studied

routing protocols in mobile opportunistic networks with realistic mobility traces. Chaintreau et al. [CHC+06] theoretically analyzed the impact of routing algorithms over a model derived from a realistic mobility data set. Su et al. [SGdL06] simulated a set of routing protocols in a small experimental network. Those studies help researchers better understand the theoretical limits of opportunistic networks and the routing protocol performance in a small network (20–30 nodes).

Deploying and experimenting large-scale mobile opportunistic networks is difficult, so we, too, resort to simulation. Instead of using a complex mobility model to mimic people's mobility patterns, we used mobility traces collected in a production wireless network at Dartmouth College to drive our simulation. Our message-generation model, however, was synthetic.

To the best of our knowledge, we are the first to simulate the effect of routing protocols in a *large-scale* mobile opportunistic network, using realistic contact traces derived from real traces of a production network with more than 5,000 users.

Using realistic contact traces, we evaluate the performance of three "naive" routing protocols (direct-delivery, epidemic, and random) and two prediction-based routing protocols, PRoPHET [LDS04] and Link-State [SGdL06]. We also propose a new prediction-based routing protocol, and compare it to the above in our evaluation.

1.3 Contributions

In this dissertation we present three major contributions:

- Wireless network users do not move randomly. Many location-prediction techniques can predict a user's next location with sufficient accuracy. Using real mobility data we simulated and compared four families of location predictors. We propose a fallback mechanism to improve the overall prediction accuracy of predictors when they

fail to make a prediction. We found that simple algorithms may perform as well as, or better than, complex algorithms. Our evaluation of location predictors with extensive Wi-Fi mobility data was first reported in Infocom 2004 [SKJH04], and later we extended this work with more predictors and more results in a paper for IEEE Transactions on Mobile Computing [SKJH06].

- We developed and evaluated several handoff-time predictors. We studied the effect of bandwidth reservation when we applied the joint time and location prediction techniques to a VoIP application [SDK$^+$06]. Although it is difficult to predict precise handoff time, even roughly accurate handoff-time predictions combined with location predictions can benefit bandwidth-reservation schemes.

- We simulated and evaluated several opportunistic network routing protocols using real traces [SK07]. We found that direct delivery, random, and epidemic routing protocols are not practical for large-scale opportunistic networks. Prediction-based routing protocols trade off message delivery ratio against resource usage. The parameters of these algorithms can be configured to suit the design goals of an opportunistic network.

In the next chapter we introduce the mobility trace that we used to drive our simulation to evaluate prediction techniques, reservation schemes and routing algorithms. In Chapter 3 we present four families of location-prediction techniques and propose several time prediction methods. We also define the metrics to compare different approaches. We analyze the performance of predictors in Chapter 4, and examine, in Chapter 5, the effect of prediction-based bandwidth reservation schemes for voice over IP applications. Chapter 6 covers the study of routing algorithms in opportunistic networks. In Chapter 7 we conclude and present some ideas for future research.

Chapter 2

Handoff Traces

In this chapter we introduce the traces that we used to drive our simulations. We first describe our data collection, discuss the limitations of our data, and the present some characteristics of the traces. At the end, we survey other traces and discuss mobility models.

2.1 Data collection

Dartmouth College has operated a campus-wide Wi-Fi (IEEE 802.11) wireless network since the spring of 2001. Although there was no specific effort to cover outdoor spaces, the campus is compact and the interior APs tend to cover most outdoor spaces.

We have monitored network usage since the network was installed. The access points transmitted a "syslog" message every time a client card authenticated, associated, re-associated, disassociated or deauthenticated; the message contained the unique MAC address of the client card. Although a given card might be used in multiple devices or a device used by multiple people, in this project we think of the wireless card as a "network user" and thus the term "user" refers to a wireless card.

The wireless population has been growing fast. As of spring 2004 there were more than

500 access points providing 11 Mbps coverage to the entire campus. There were between 2,500 and 3,500 users active on most school days. After June of 2004, the Dartmouth College's Wi-Fi network made a dramatic change: it replaced its more than 500 old access points with more than 1500 new access points. There were about 4,500 users on most school days in Spring 2006, while the total student enrollment of Fall 2005 was 5,780 (4,110 were undergraduates). There were also about 1,000 faculty members and 3,000 staff.

2.2 Challenges of our data

We used the data set that was collected between April 2001 and June 2004 [KHA04]. Because of the dramatic change of the network after June 2004, the newer data set, which was collected after June 2004, was not yet cleaned up. Even the data that we used had some brief "holes" when the campus experienced a power failure, or when a central syslog daemon apparently hung up. Most of the holes were shorter than 5 minutes and frequently happened during the first months of network installation. There were only 16 holes longer than 5 minutes during 2004. The longest hole, however, was 44 hours. Also, since syslog uses UDP it is possible that some messages were lost or mis-ordered. In March 2002, we installed two redundant servers to record the data, so that holes in one of the servers can be filled by the data recorded by the other server. In October 2003, we moved our server to the machine room for our wireless network and removed the redundant server.

The AP firmware had several bugs in the first year. In particular, the syslog entries were sometimes produced using uninitialized strings.

After we cleaned the data of some of these glitches and merged the data from two servers (where available), we extracted user traces from the data. Although there were many holes in the early collection of our traces, there were relatively few users. Therefore,

we did not treat the holes specially.

Each user's trace is a series of location and time pairs, that is, access-point name and the time. A user's traces do not necessarily represent physical moves, because the wireless card will sometimes associate and re-associate with a number of nearby APs, even while the user remains stationary.

We introduced the special location "OFF" to represent the user's departure from the network (which occurs when the user turns off their computer or their wireless card, or moves out of range of all access points). It was challenging to detect the "OFF" state. We found that in our syslog, some cards issued disassociation messages, but most cards did not. For most cards, the AP eventually issues a deauthentication message when the card has been inactive for 30 minutes. Thus, we define OFF to have occurred at a card's disassociation message or after deauthentication by the last-associated AP. Deauthentications may be issued by an AP for several different reasons. In those cases where "Inactivity" was the reason, which occurs when the AP detects no activity from the card for 30 minutes, we set the time of OFF state to 30 minutes prior the time of the deauthentication message. In all other deauthentication cases, we set the time of the OFF state to the time of the deauthentication message.

There are three ways to treat the OFF state when doing location prediction. First, we could treat OFF in exactly the same way as any other symbol (access-point name) in the users' traces. Second, we could treat OFF as a transparent state, which means that the user moves from the AP before OFF to the AP after OFF directly in one step. We used the first approach because we think that predicting OFF is useful for some applications. In our joint time and location prediction evaluation (Section 4.2), however, we did not count the predictions to and from OFF, because the metrics, over-provisioning and under-provisioning, do not make sense of OFF.

2.3 Characteristics

Now, we discuss some characteristics of the traces.

2.3.1 Trace length and visit distribution

First, we examined each user's trace length, which is the number of location changes (including OFF). We selected two years of traces that were collected from the beginning of the network, April 2001 to March 2003. The traces varied widely in length (number of locations in the sequence), with a median of 494 and a maximum of 188,479; most are shown in Figure 2.1. Users with longer traces were either more active (using their card more), more mobile (thus changing access points more often), or used the network for a longer period (some users have been on the network since April 2001, and some others only later arrived on campus). Some users visited many APs during the period of measurement, while some others only visited a small number of APs. Figure 2.2 shows the distribution of the number of visits for each user. It indicates that most users visited relatively few APs; about 80% visited about 50 APs or fewer. We are also interested in the density of visits per AP for each user. Some users visited many APs, but only a few visits for each AP. However, some other users intensively visited a small number of APs. Figure 2.3 presents the distribution of average visits per AP for each user. It indicates that most users have relatively few visits to any given AP; about 80% had about 50 visits or fewer per AP. These curves illustrate the potential difficulty of prediction; while users visit relatively few APs, the history of their movement from a given AP is relatively small.

2.3.2 Duration

Wireless users may not always be associated with the network, even in cellular networks where cell phones are expected to be always on. Users may turn off their devices, and users

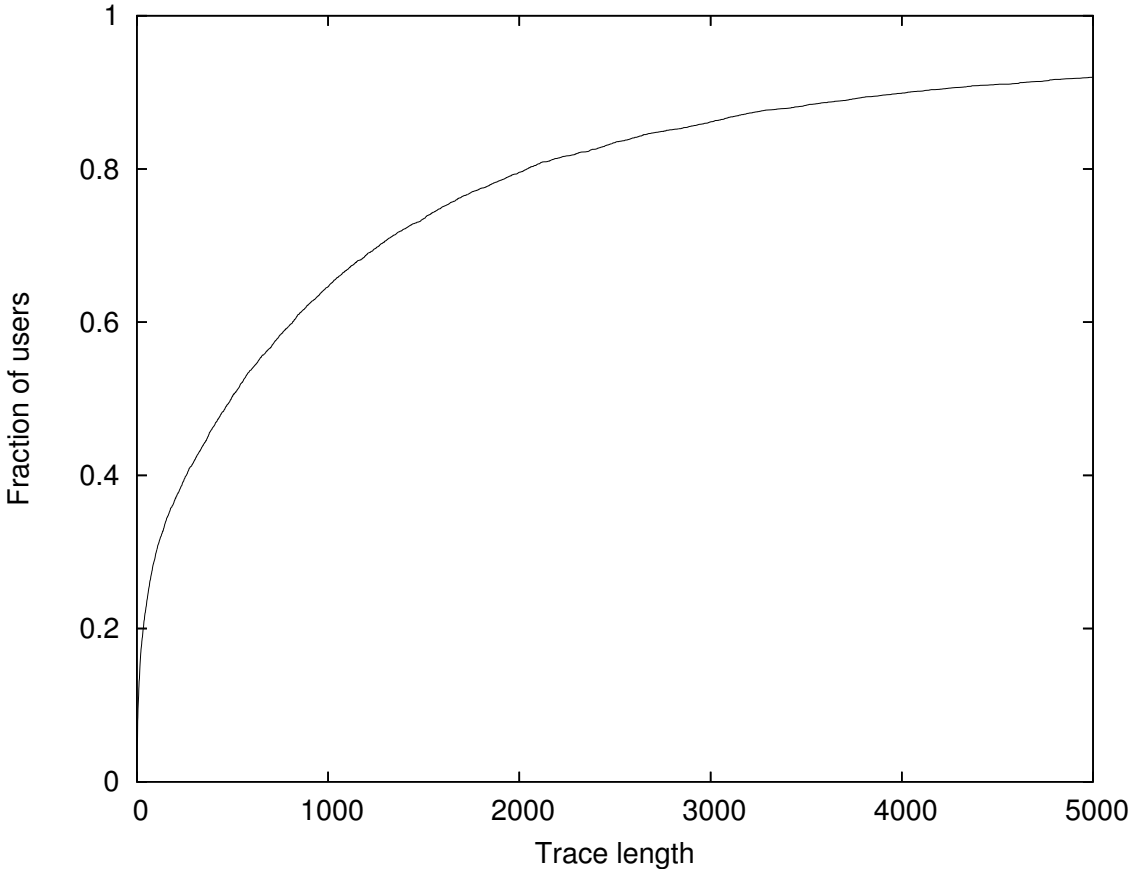


Figure 2.1: Length of user traces

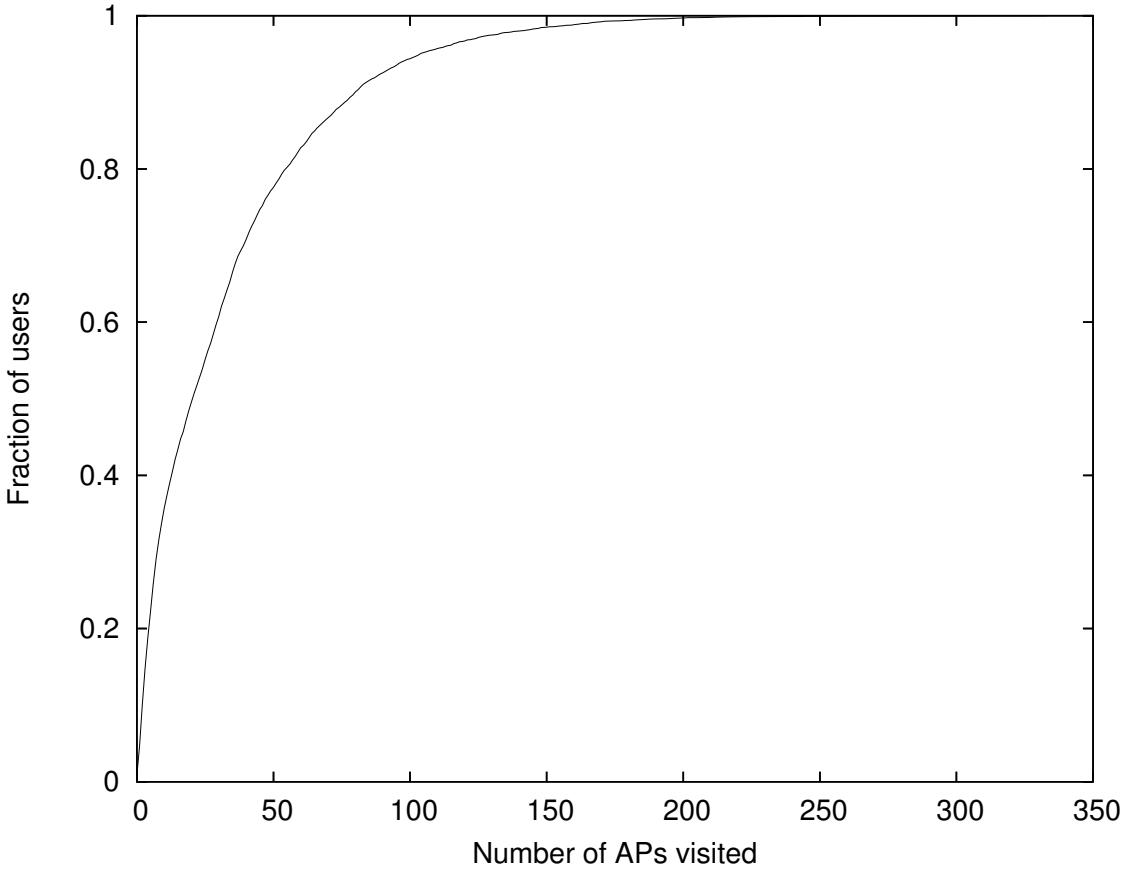


Figure 2.2: Number of visits

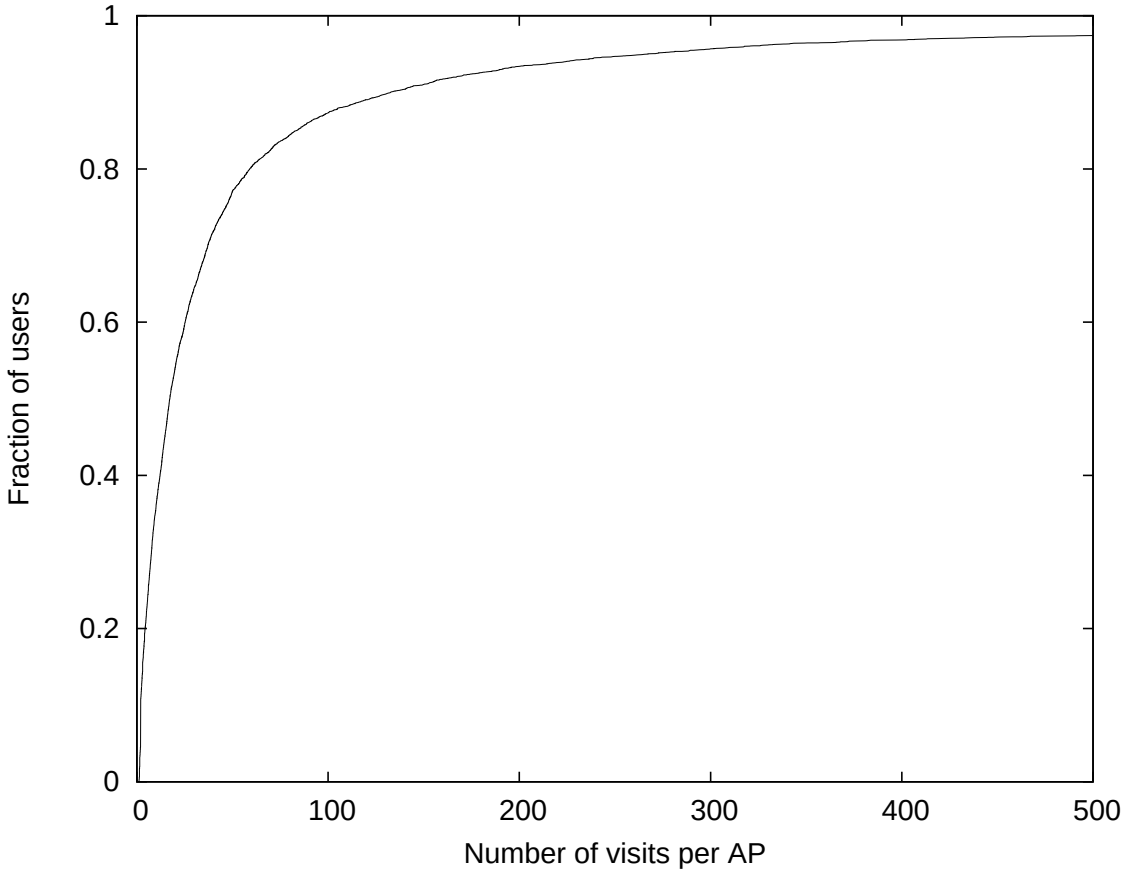


Figure 2.3: Visits per AP

may move out of the radio coverage. We select six months of data when the network was mature. The data were collected between January 1, 2004 and June 30, 2004. We measure how long a user stayed at an AP, how long a user stayed connected, and how long a user was off the network.

We define a *session* to be the time when a user connects to the network until the time that the user next disconnects from the network (moved to "OFF"). A session may visit several APs.

Figure 2.4 shows that all three durations had a similar distribution. The plot was cut off at one-hour duration. More than 85% of all sessions lasted less than one hour. The AP durations were generally shorter than than the session durations, since a session often included more than one AP. The OFF duration curve crosses the other two curves, which means the OFF duration had a large spread: many short OFF durations, as well as many long OFF durations. A 6-minute length counts for 80%, 77%, and 69% of AP, OFF, and session durations, respectively. The short session and OFF durations mean that users switch often on and off the wireless network. The short AP durations may be caused by the wireless cards that frequently change their access points.

2.3.3 Device types

Since our wireless network has a variety of devices, such as laptop computers, personal digital assistants (PDAs), and voice over IP (VoIP) devices, we expected that different type of devices have different characteristics. The device types were classified by Tristan Henderson [HKA04]. The types of device were identified by their operating system in use, or an Organizationally Unique Identifier (OUI) for special devices. Device types include

- **Windows:** Windows laptop;
- **MacOS:** Apple laptop;

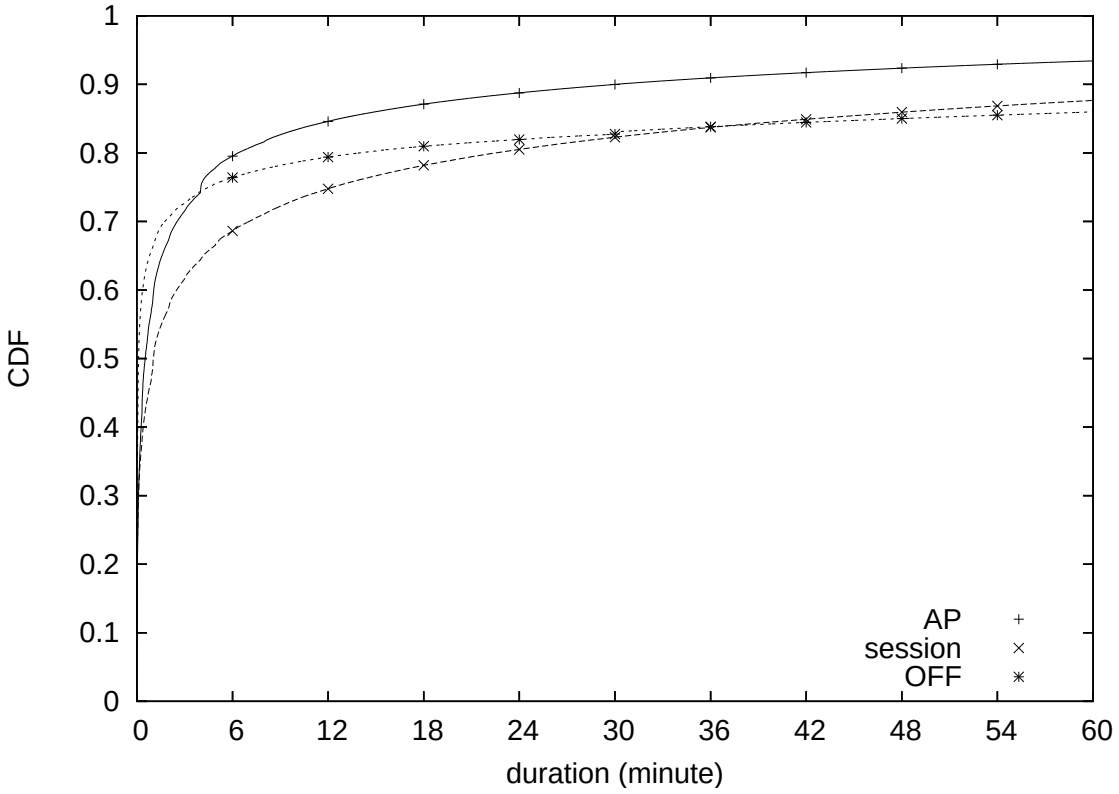


Figure 2.4: Distribution of durations for all users. The AP durations are generally shorter than than the session durations. The OFF duration curve crosses the other two curves, which means the OFF duration had a large spread.

- **Linux:** Linux laptop;
- **DualBoot:** Windows/Linux dual bootable laptop;
- **PocketPC:** PDA with PocketPC OS;
- **PalmOS:** PDA with PalmOS;
- **Cisco phone:** VoIP device;
- **Vocera:** Vocera badge - VoIP device;
- **Unknown:** devices that we cannot identify.

We assume that VoIP devices – Cisco phones and Vocera badges – are usually always-on, while other devices such as laptop computers or PDAs tend to switch off when they move (we call the latter on-and-off devices).

Figure 2.5 shows the distribution of durations that a user stayed connected to the network for each type of device. Linux and Windows/Linux dual bootable laptops have longer session durations than other type of devices. The reason may be that most of the Linux and Windows/Linux laptops are used by computer science department and engineering school students and researchers, and many of them have been used for wireless network experiments so that they stay longer for a session. The PalmOS PDAs have the shortest session duration. We think that those PDAs turn off after a short use to save battery life. However, the PocketPC PDAs show similar distributions as Windows and MacOS laptops. To our surprise, the VoIP devices—Cisco phones and Vocera badges—did not behave much differently in session duration from Windows and MacOS laptops, because the Cisco phones and Vocera badges are always-on devices, we were expecting different behavior between those always-on devices and the on-off devices (laptop computers). And this similarity also support the validity of our use of the trace for simulation of voice applications (Chapter 5).

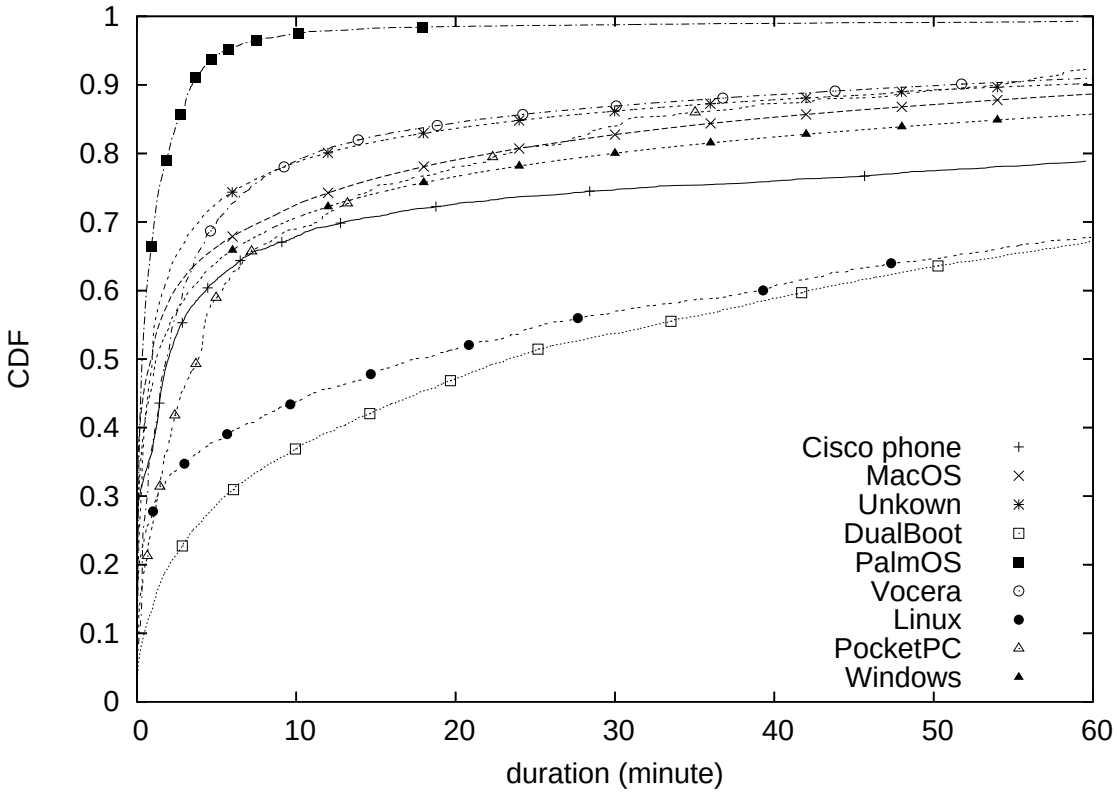


Figure 2.5: Session duration distributions for different type of devices

The distribution of OFF durations of devices (Figure 2.6) shows broad differences. Notice there is a big jump at 1800 seconds (half hour) of the curves for Linux laptop and Cisco phone. We suspect that jump is caused by some false OFF detection—when we do not receive the disassociation messages, we consider the devices as OFF at the time a half hour earlier than the deauthentication messages of "Inactivity." The devices might not in fact go off, but the network detected them right after the "Inactivity" message. However, we counted an exact 30-minute OFF duration. PocketPC PDAs, DualBoot laptops and PalmOS devices are among the longest OFF-duration devices. We suspect those devices are not often used for wireless networking. Vocera badges had the shortest OFF duration, because they use the Wi-Fi network extensively.

2.3.4 OFF state

We studied how often a device went OFF after roaming among APs, that is, after how many AP changes the device went OFF. This factor is important in the design of association models.

We define the OFF-state rate as the inverse of the number of consecutive APs plus one OFF state. For example, a 0.5 OFF-state rate includes all traces consisting of one AP followed by an OFF; a 0.1 OFF-state rate is those traces in which an OFF follows an average of 9 consecutive other APs. Figure 2.7 shows distribution of the OFF-state rate of all devices. There were a significant number of devices that had near half OFF-state rate. Those devices rarely roam among APs. Since most of the devices in our wireless network are laptops, we suspect that users did not carry their laptops around while the laptop was associated with an AP. Users usually close the laptops when they move.

For VoIP devices, they are convenient to use when the users are moving. Figure 2.8 clearly shows that Cisco phones, Vocera badges and PocketPC had lower OFF-state rates.

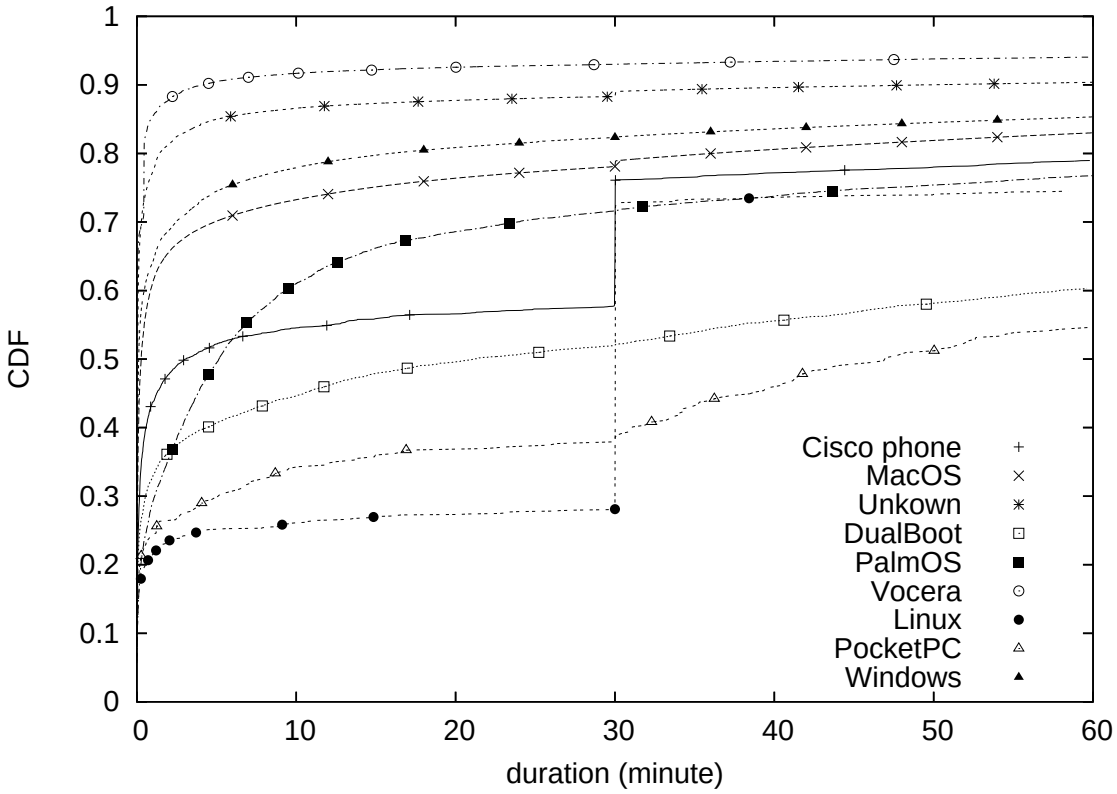


Figure 2.6: OFF duration distributions for different type of devices

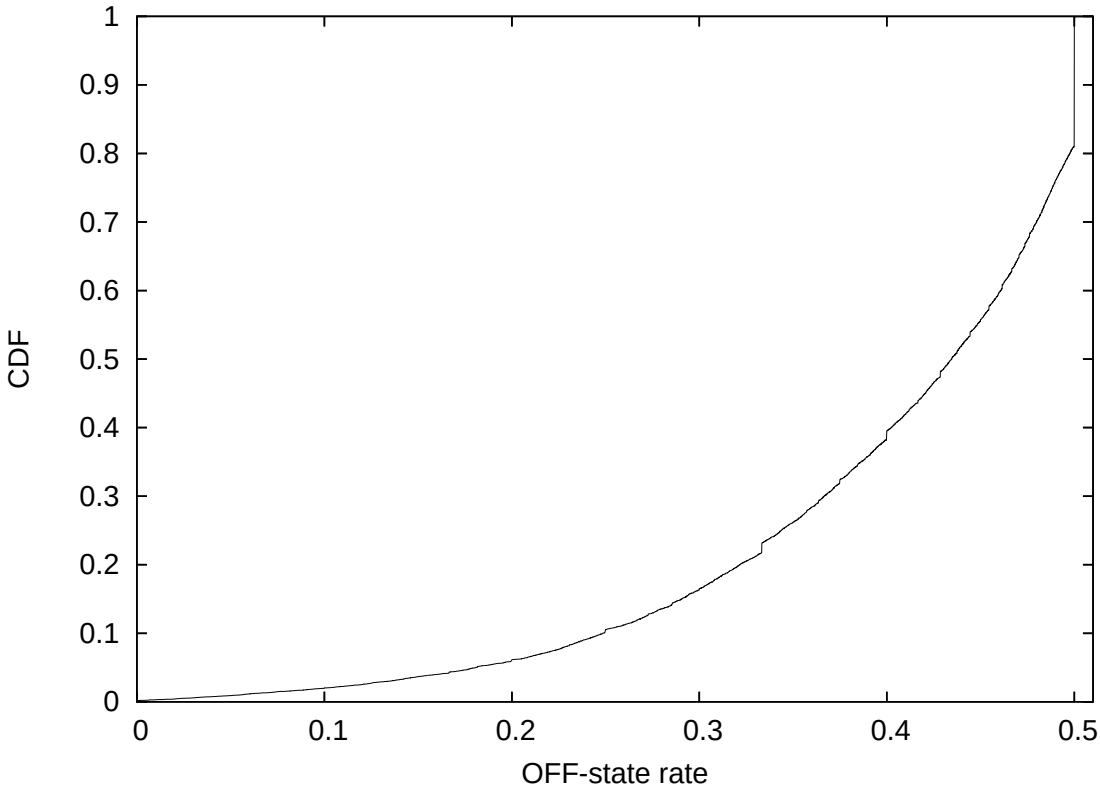


Figure 2.7: OFF-state rate of all devices (CDF).

These device are easy to carry around, and people do use them while walking, especially the phones and Vecera badges. In contrast, the PalmOS devices had higher OFF-state rate, which matches the short session durations in Figure 2.5.

2.4 Other wireless network trace studies

Before the proliferation of wireless local-area networks, there were few studies of real wireless users because it was (and is) difficult to get cellular network providers to supply traces. When the Wi-Fi networks became popular, many researchers collected wireless traces and studied the characteristics of their networks. Most of these traces are available at the CRAWDAD website [CRA].

Tang and Baker carried out the first large-scale wireless-network usage and mobility study [TB02]. They studied the Metricom metropolitan-area network with more than 25,000 users for seven weeks. They examined when and where the network was most active, and how often and how far users moved. They found that the locations that a user visits on a daily basis were the locations that were close to each other. Later, they studied a building-wide local-area network [TB00]. This time they focused more on user traffic than on user mobility. The population of their study was people in their department of computer science.

Kotz and Essien [KE05] analyzed Dartmouth's campus-wide network with 476 APs and more than 1,700 users during seven weeks. They found that some wireless cards were unable to settle on one AP, and hence those cards roamed excessively. Later, Henderson, Kotz and Abyzov [HKA04] studied the changing usage of the Dartmouth wireless network for a seventeen-week period. This time the network included more than 500 APs and 7,000 users. They introduced "session diameter" to show that there were different mobility characteristics for different devices. They found that users were surprisingly non-mobile,

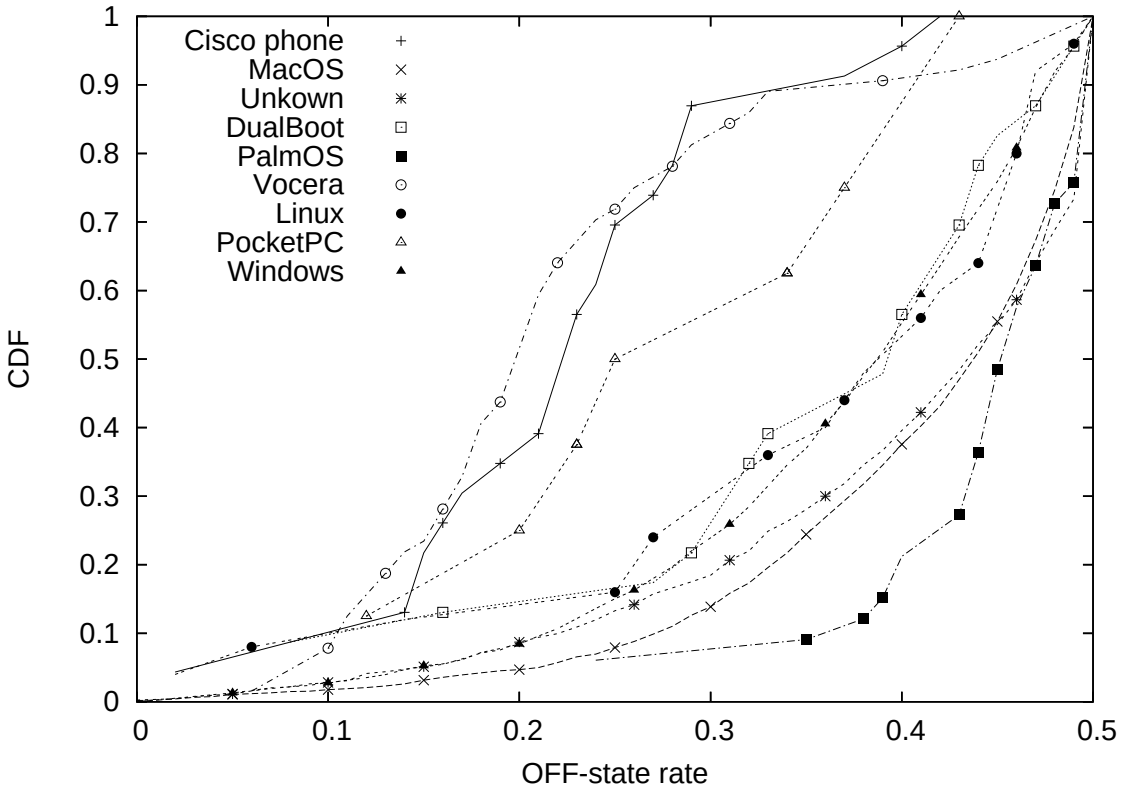


Figure 2.8: OFF-state rate of all devices by device type.

with half remaining close to home (defined as spending more than 50% of their time at one location) about 98% of the time.

Balazinska and Castro [BC03] analyzed a corporate wireless LAN workload of 1,366 users across 177 access points. They introduced two metrics to characterize the wireless user mobility, namely, *prevalence* and *persistence*. Their finding agrees with Dartmouth's traces [HKA04] that many users spend a large fraction of their time at a single location.

Schwab and Bunt [SB04] characterized the usage of a campus wireless network at the University of Saskatchewan with a week-long traffic trace. They analyzed the roaming degree and the distribution of the number of access points visited by each unique user. They found that an average user only connected to a limited number of access points during a week.

McNett and Voelker [MV05] analyzed the mobility patterns of users of wireless handheld PDAs in UCSD's campus wireless network using an eleven-week trace of wireless network activity. They developed two wireless network topology models for use in wireless mobility studies: an evolutionary topology model based on user proximity and a campus waypoint model that serves as a trace-based complement to the random waypoint model.

Chinchilla, Lindsey, and Papadopouli [CLP04] studied the performance of different caching paradigms in the wireless infrastructure network at the University of North Carolina at Chapel Hill. They used a Markov method to predict a wireless user's next AP and achieved high accuracy (86%). Later, Papadopouli, Shen, and Spanakis [PSS05], extending the previous work, characterized and analyzed the wireless access patterns based on mobility, session and visit durations. They found that mobility and building type affect the session and visit durations.

We focused on the mobility of human pedestrians. Some researchers are interested in studying the mobility of wild animals. ZebraNet [JOW+02] collected the mobility traces of zebras in Africa, and studied their mobility patterns.

2.5 Mobility models

Wireless networks encourage mobility. Wireless users' mobility patterns change over time and differ from user to user. New protocols or topologies may be required to accommodate the unusual characteristics of a wireless network. Due to the difficulty of deploying and configuring of real research networks, most network protocol and design evaluations are carried out through simulations. A realistic network model is important to such simulations. A wireless network model usually consists of three sub-models: a user mobility model, a radio propagation model, and a data traffic model. The user mobility model and the radio propagation model together determine the connectivity of mobile users in the simulation. The traffic model is used to generate data traffic over the network.

The Random Walk Model, Random Waypoint Model [JM96], and Random Direction Model [RMSM01] are the most often used for mobile wireless-network simulations. Many researchers believe that simulation results obtained with unrealistic movement models might not correctly reflect the true performance of the network protocols. Kurkowski, Camp, and Colagrosso [KCC05] surveyed 151 papers from MobiHoc and challenged the credibility of many papers. Credible simulation results rely on realistic mobility models or real traces. As one step towards realistic mobility models, Jardosh et al. [JBRAS03] incorporated obstacles to the random walk models. Bai, Narayanan and Helmy [BNH03] propose a city model and a freeway model to constrain users mobility within streets and roads. These improved mobility models, however, still assume that users move randomly. For more mobility models, please refer to Camp et al.'s survey [CBD02].

All the models surveyed by Camp et al. are based on the assumption that the user will move randomly, possibly with constraints. These models set up a fixed number of users who randomly move within the simulated area. In real networks, however, new users join the network, while current users may leave (move out of the coverage of the network or

simply disconnect from the network by turning off their wireless devices). In sensor networks, sensors may stop functioning and new sensors may be deployed. Furthermore, users never move truly randomly. None of the existing models takes this reality into account.

Many researchers study the wireless user mobility traces to build realistic mobility models. Bhattacharjee et al. [BRSS04] developed a hybrid mobility model by physically observing the actual movement patterns of people on campus and the post-processing the data to form their spatio-temporal distributions. They claim that their method differs from the synthetic and wireless trace-based models in that their model is empirical in nature and devoid of the restrictions as observed in some of the existing mobility models. Kim and Kotz [KK05a] used the Fourier transform to convert time-dependent user location information to the frequency domain and used two strong periods as parameters to classify users and APs based on Bayesian theory. They found that both users' and APs' behavior had a strong period of one day. Most users had one day as their primary period and a smaller secondary period. Later, Kim and Kotz proposed a mobility model [KK05b] based on their findings from the Dartmouth wireless network. Jain, Lelescu, and Balakrishnan [JLB05] studied the Dartmouth traces and found that user-registration patterns (handoff patterns) exhibit a distinct hierarchy and that WLAN access points can be clustered based on registration patterns. They propose a registration model that generates synthetic traces. They compared the synthetic traces against the corresponding traces from the test set, finding that the synthetic traces agree well with the test set. They showed that a simple modified random waypoint model does not represent the real data.

Instead of using a complex mobility model to mimic people's mobility patterns, we used mobility traces collected at Dartmouth College to drive our simulations.

Chapter 3

Predicting Handoff

Handoff prediction is the process of anticipating a roaming user's next handoff event, which includes where (location) and when (time) the user will move. In this chapter, we first define location, then present several location prediction techniques that predict the destination of a handoff will happen. Next, we introduce time prediction that estimates the handoff time. Combining the time predictor and location predictor, we get a full predictor. We also discuss the metrics that we used to evaluate the performance of our prediction algorithms, and define entropy to study the coefficient correlations between handoff traces and location prediction accuracy. At the end of this chapter, we survey other handoff prediction techniques and discuss their applications.

3.1 Location

In the context of this dissertation we assume that a user resides at a given discrete location at any given time; sample locations include "room 116" within a building, "Thayer Dining Hall" within a campus, "cell 24" within a cellular network, or "berry1-ap" access point within an 802.11 wireless network.

In our data, as we discuss in Chapter 2, we have available the name of the access point with which the user's device is associated. We also have the name of the buildings where the access points were located (or the nearest buildings). For simplicity we call the access point where the user's device is registered a *location*, although this should not be taken to imply that we know the user's precise geographical location in, say, latitude and longitude.

We examine prediction algorithms that use only symbolic locations. Given the limitation of our data, it is not trivial to extract user's physical mobility, and we do not use speed and direction to compute the next possible time and location. Instead, we predict the next location, or compute the probability of a handoff within a period of time, using a history of prior handoffs.

We list all possible locations in a finite alphabet $\mathcal{A}$, and can identify any location as a symbol a drawn from that alphabet. For a given user we list the sequence of locations visited, its *location history* L, as a string of symbols. If the history has n locations, $L = a_1 a_2 \ldots a_n$, and $a_i \in \mathcal{A}$, for $1 \leq i \leq n$.

The location history may be a sequence of location *observations*, for example, the user's location recorded once every five seconds, or a sequence of location *changes*. In the latter case, $a_i \neq a_{i+1}$ for all $1 \leq i < n$. All of the predictors we consider in this dissertation are agnostic to this issue. Our data are a sequence of location changes, accompanied with the time when the location changed. We refer to a location change as a *move*.

All of the predictors we consider here are domain-independent and operate on the string L as a sequence of abstract symbols. That is, they do not place any interpretation on the symbols. For that reason, we ignore any timing information in the location history, and do not require any associated information relating the symbols, such as geographic coordinates. As an example, though, consider the environment with six wireless cells (labeled b through g) diagrammed in Figure 3.1: one possible location history can be $L = gbdcbgcefbdbde$. Therefore, in this chapter, the names *location* and *symbol* are

interchangeable.

3.2 Prediction algorithms

We consider four families of location predictors, two approaches to time prediction, and one combined time and location handoff predictor.

3.2.1 Markov predictors

The order-k (or "$O(k)$[1]") Markov predictor assumes that the location can be predicted from the current *context*, that is, the sequence of the k most recent symbols in the location history. The underlying Markov model represents each state as a context, and transitions represent the possible locations that follow that context.

Consider a user whose location history is $L = a_1 a_2 \ldots a_n$. Let substring $L(i,j) = a_i a_{i+1} \ldots a_j$ for any $1 \leq i \leq j \leq n$. We think of the user's location as a random variable X. Let $X(i,j)$ be a string $X_i X_{i+1} \ldots X_j$ representing the sequence of random variable $X_i, X_{i+1}, \ldots X_j$ for any $1 \leq i \leq j \leq n$. Define the context $c = L(n-k+1, n)$. The Markov assumption is that X behaves as follows, for all $a \in \mathcal{A}$ and $i \in \{1, 2, \ldots, n-k+1\}$:

$$\begin{aligned} &P(X_{n+1} = a \mid X(1,n) = L) \\ &= P(X_{n+1} = a \mid X(n-k+1, n) = c) \\ &= P(X_{i+k} = a \mid X(i, i+k-1) = c), \end{aligned}$$

where the notation $P(X_i = a_i \mid \ldots)$ denotes the probability that X_i takes the value a_i. The first two lines indicate the assumption that the probability depends only on the context of

[1]Note that here our notation $O(k)$ is different from traditional "Big-O" notation. Here, the order k means that the length of the context is k, while traditional "Big-O" notation does not mean exact k.

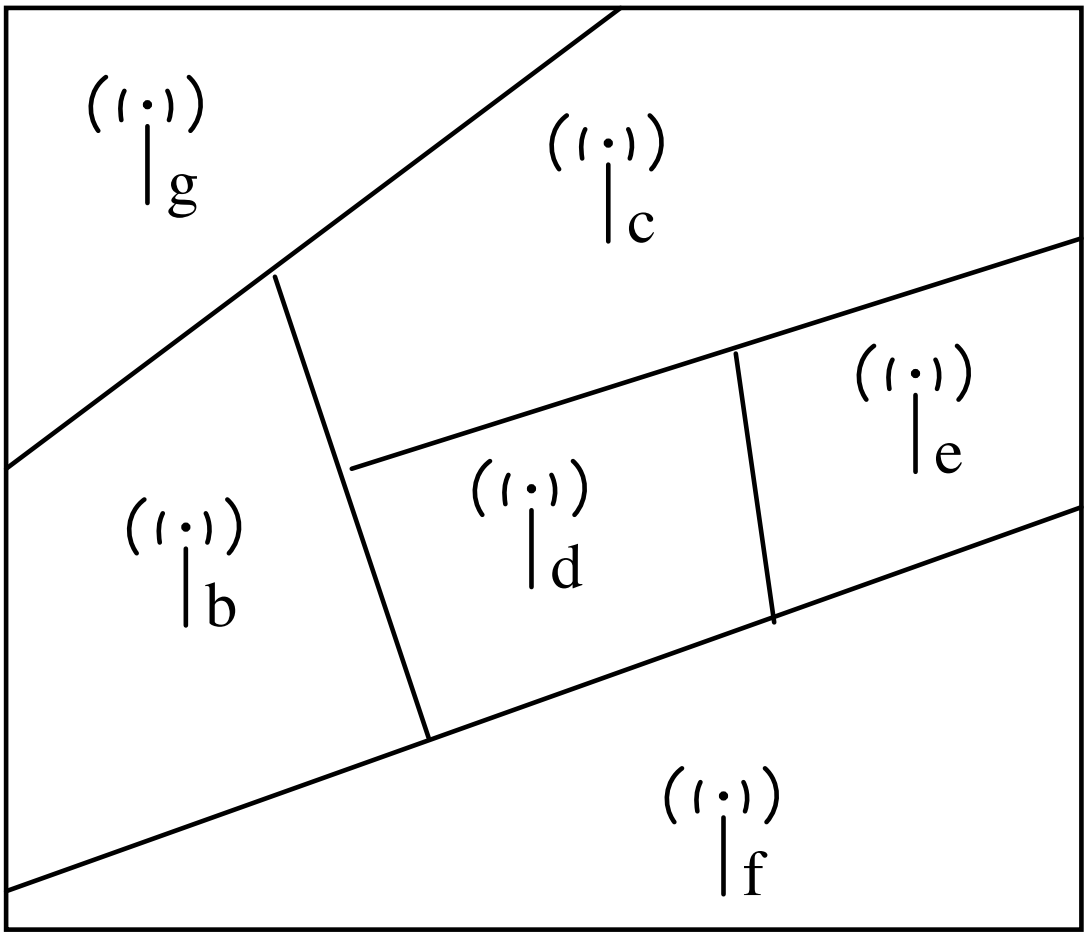


Figure 3.1: Sample cell map with six cells

the k most recent locations. The latter two lines indicate the assumption of a stationary distribution, that is, that the probability is the same when the context is the same.

These probabilities can be represented by a *transition probability matrix* M. Both the rows and columns of M are indexed by length-k strings from $\mathcal{A}^k$ so that $P(X_{n+1} = a \mid X(1,n) = L(1,n)) = M(s,s')$, where $s = L(n-k+1,n)$, the current context, and $s' = L(n-k+2,n)a$, the next context. In that case, knowing M would immediately provide the probability for each possible next symbol of L.

Since we do not know M, we can generate an estimate $\widehat{P}$ from the current history L, the current context c of length k. The probability for the next symbol to be a is

$$P_k(a) = \widehat{P}(X_{n+1} = a \mid L) = \frac{N(ca, L)}{N(c, L)} \tag{3.1}$$

where $N(s', s)$ denotes the number of times the substring s' occurs in the string s.

Given this estimate, we can easily define the behavior of the $O(k)$ Markov predictor. It predicts the symbol $a \in \mathcal{A}$ with the maximum probability $\widehat{P}(X_{n+1} = a \mid L)$, that is, the symbol that most frequently follows the current context c in prior occurrences in the history. Notice that if c has never occurred before (the only occurrence is the current context), the above equation evaluates to $0/1 = 0$ for all a, and the $O(k)$ Markov predictor makes no prediction.

If the location history is not generated by an $O(k)$ Markov source, then this predictor is, of course, only an approximation.

Fortunately, $O(k)$ Markov predictors are easy to implement. Our implementation maintains an estimate of the (sparse) matrix M, using Equation 3.1. For simplicity, we used the occurrence frequency as the value for each entry in the matrix M: we store $N(ca, L)$ in each entry of row c, rather than $N(ca, L)/N(c, L)$. To make a prediction, the predictor scans the row of M corresponding to the current context c, choosing the entry with the

highest probability for its prediction. After the next handoff occurs, the predictor updates the appropriate entry in that row of M, and updates c, in preparation for the next prediction.

Our locations are symbols, and the histories or contexts are strings of symbols. We operate on a string using a procedure SUBSTR(s, i, n). This procedure returns a substring, which starts at index i (the index of the first symbol is 0) of the string s, and continues up to n symbols. Thus, when LENGTH$(s) - i \geq n$, the substring is of length n; otherwise, it is of length LENGTH$(s) - i$.

Here, we list the procedure MARKOV-PREDICTION that predicts the next symbol using a length-k context string c and the Markov transition matrix M after it arrives at a new symbol p. Each row of the matrix M is a list of symbol and value pairs (p, v).

MARKOV-PREDICTION(M, c, p, k)

```
1  if LENGTH(c) < k
2     then
3          c ← c + p                ▷ append p to c
4          return NIL
5  Q ← M_c                          ▷ find a reference to the row of c in Matrix M
6  Q_p ← Q_p + 1                    ▷ update the entry (c, p) of matrix M
7  c ← c + p                        ▷ append p to c
8  c ← SUBSTR(c, 1, k)              ▷ update the context c
9  Q ← M_c                          ▷ find the row of new c in matrix M
10 return FINDMAX(Q)
```

When there is no such row of c in matrix M, the row Q is empty. When a pair (p, v) does not exist in a row Q, the default value of Q_p is 0. Now, we present the procedure FINDMAX, which finds the symbol p such that v has the maximum value, for a given list Q of pairs (p, v).

FINDMAX(Q)

1 $v_x \leftarrow 0$ ▷ initialize the maximum value

2 $p_x \leftarrow$ NIL ▷ initialize the symbol with maximum value

3 **foreach** pair (p, v) in Q

4 **do if** $v > v_x$ ▷ check the maximum value

5 **then** $v_x \leftarrow v$ ▷ update the maximum value

6 $p_x \leftarrow p$ ▷ update the symbol with maximum value

7 **return** p_x

3.2.2 Fallback

Notice the above procedure may fail to find an exact match for the order-k context c, so the predictor predicts NIL. We introduce a meta-predictor based on the Markov predictor family, using a technique we call *fallback*. The fallback mechanism works this way: when an $O(k)$ Markov predictor fails to make a prediction, we use an $O(k-1)$ Markov predictor to predict the next location. In general, the $O(k)$ fallback predictor recursively uses the result of the $O(k-1)$ predictor (with $k = 0$ as the base of the recursion) whenever it encounters an unknown context. We define the $O(0)$ Markov predictor as one that ignores the current context and returns the most frequently visited location as a prediction.

3.2.3 LZ-based predictors

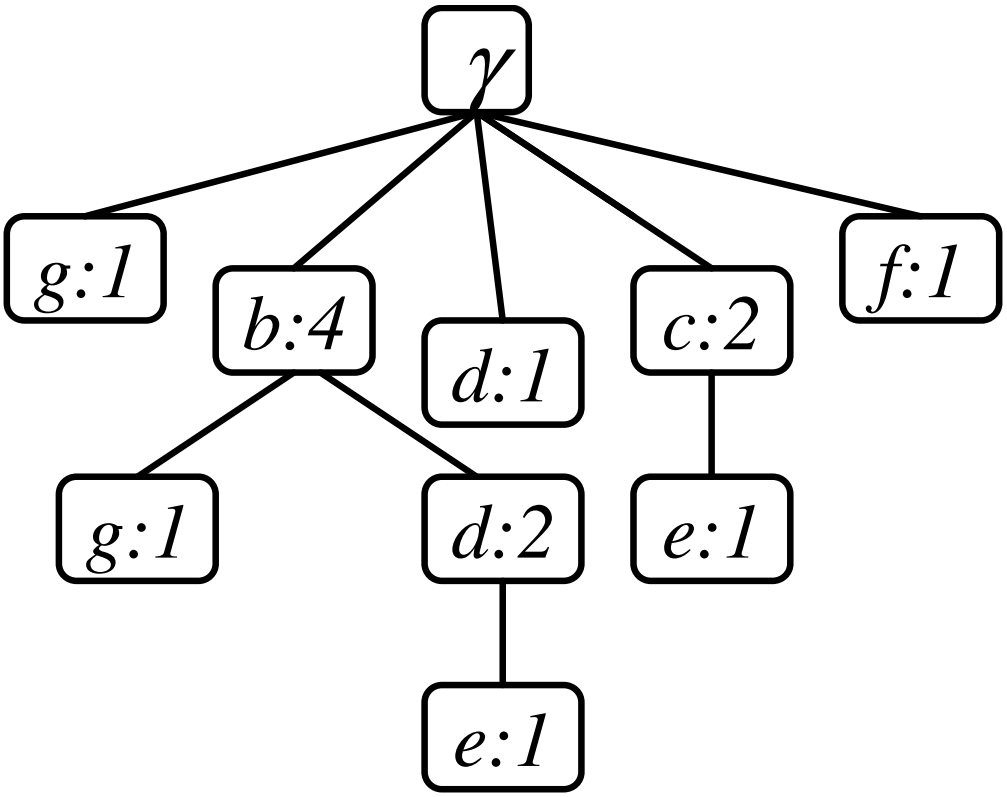
LZ-based predictors are based on a popular incremental parsing algorithm by Ziv and Lempel [ZL78] often used for text compression. This approach seems promising because (a) most good text compressors are good predictors [VK96] and (b) LZ-based predictors are similar to the $O(k)$ Markov predictor except that k is a variable allowed to grow to infinity [BD02]. We briefly discuss how the LZ algorithm works.

Let γ be the empty string. Given an input string s, the LZ parsing algorithm partitions the string into distinct substrings $s_0, s_1, \ldots, s_m$ such that $s_0 = \gamma$, and for all $j > 0$ substring s_j without its last character is equal to some $s_i, 0 \leq i < j$, and $s_0 s_1 \ldots s_m = s$. Observe that the partitioning is done sequentially, i.e., after determining each s_i the algorithm only considers the remainder of the input string. For example, the string $s = gbdcbgcefbdbde$ is parsed to $\gamma, g, b, d, c, bg, ce, f, bd, bde$.

Associated with the algorithm is a tree, the *LZ tree*, that is grown dynamically during the parsing process. Each node of the tree represents one substring s_i. The root is labeled γ and the other nodes are labeled with a symbol from $\mathcal{A}$ so that the sequence of symbols encountered on the path from the root to that node forms the substring associated with that node. Since this LZ tree is to be used for prediction, it is necessary to store some statistics at each node. The tree resulting from parsing the above example is shown in Figure 3.2; each node is labeled with its location symbol and the value of its statistic, a counter, after parsing.

To parse a (sub)string s we trace a path through the LZ tree. If any child of the current node (initially the root) matches the first symbol of s, remove that symbol from s and step down to that child, incrementing its counter; continue from that node, examining the next symbol from s. If the symbol did not match any child of the current node, then remove that symbol from s and add a new child to the current node, labeled with that symbol and counter=1; resume parsing at the root, with the now shorter string s.

Vitter and Krishnan [VK96] considered the case when the generator of L is a finite-state Markov source, which produces sequences where the next symbol is dependent on only its *current state*. (We note that a finite-state Markov source is different from the $O(k)$ Markov source in that the states do not have to correspond to strings of a fixed length from $\mathcal{A}$.) The



$$s = gbdcbgcefbdbde$$

$$s_i = \gamma, g, b, d, c, bg, ce, f, bd, bde$$

Figure 3.2: Example LZ parsing tree

probability, for each $a \in \mathcal{A}$, can be estimated as

$$\widehat{P}(X_{n+1} = a \mid L) = \frac{N^{LZ}(s_m a, L)}{N^{LZ}(s_m, L)} \tag{3.2}$$

where $N^{LZ}(s', s)$ denotes the number of times s' occurs as a prefix among the substrings $s_0, \ldots, s_m$ which were obtained by parsing s using the LZ algorithm.

It is worthwhile comparing Equation (3.1) with Equation (3.2). While the former considers how often the string of interest occurs in the entire input string, the latter considers how often it occurs in the partitions s_i created by LZ. Thus, in the example of Figure 3.2, while *dc* occurs in L it does not occur in any s_i.

The LZ predictor chooses the symbol a in $\mathcal{A}$ that has the highest probability estimate, that is, the highest value of $\widehat{P}(X_{n+1} = a \mid L)$. An on-line implementation needs only build the LZ tree as it parses the string, maintaining node counters as described above. After parsing the current symbol the algorithm rests at a node in the tree. It examines the children of the current node, if any, and predicts the symbol labeling the child with the highest counter, which indicates the highest frequency of past occurrence. If the current node is a leaf, the LZ predictor makes no prediction.

Each LZ-based predictor maintains an LZ-tree T and the current tree node n, which corresponds to the current string s. Each node n has two properties: the symbol n_p and the value n_v. After a user arrives at a location (symbol) p, it predicts the next symbol. We present the algorithm LZ-PREDICTION below.

LZ-PREDICTION(n, p)

```
1  Q ← all children of node n
2  foreach node q in Q
3      do if q_p = p
4          then q_v ← q_v + 1            ▷ update the value of the matching node
5               n ← q                    ▷ move the current node to q
6               return FINDMAXCHILD(q)
7  add a new child q to node n
8  q_p ← p                               ▷ set the symbol of node q to p
9  q_v ← 1                               ▷ set the value of q to 1
10 n ← γ                                 ▷ set the current node to the root
11 return NIL
```

The procedure FINDMAXCHILD finds the symbol of a node n's child that has the maximum value.

FINDMAXCHILD(n)

```
1  Q ← (p, v) pairs of node n's children
2  return FINDMAX(Q)
```

3.2.4 LZ+Prefix+PPM

Bhattacharya and Das propose a heuristic modification to the construction of the LZ tree [BD02], as well as a way of using the modified tree to predict the most likely cells that contain the user so as to minimize paging costs to locate that user.

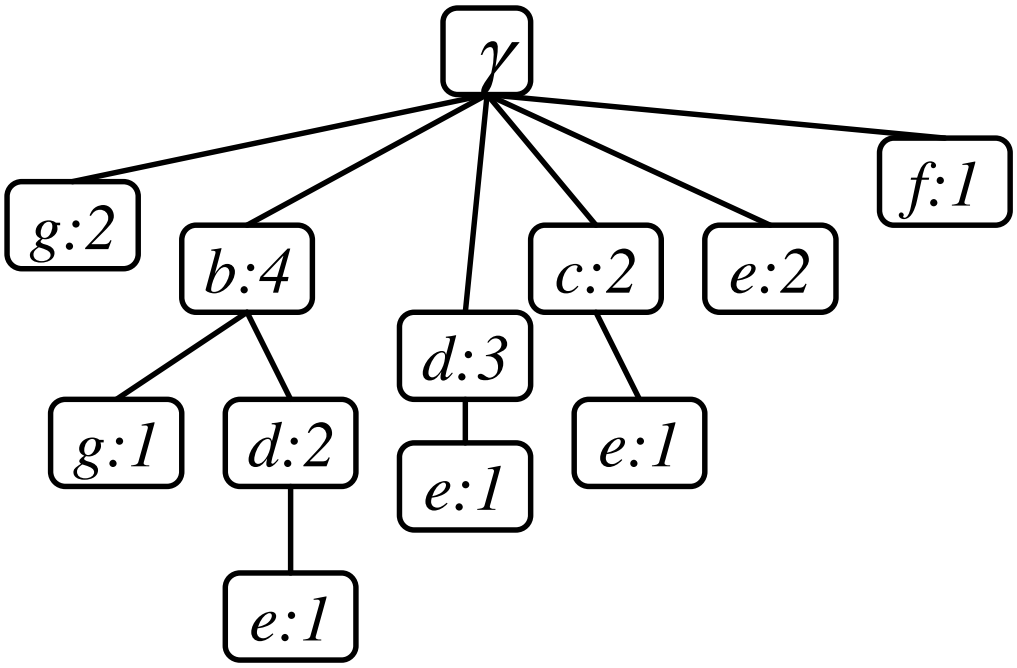
As we mention above, not every substring in L forms a node s_i in the LZ parsing tree. In particular, substrings that cross boundaries of the s_i, $0 < i \leq m$, are missed. Further, previous LZ-based predictors take into account only the occurrence statistics for the prefixes of

the leaves. To overcome this limitation, they proposed the following modification. When a new leaf is created for s_i, all the proper suffixes of s_i (i.e., all the suffixes not including s_i itself) are also inserted into the tree. If a node representing a suffix does not exist, it is created, and the occurrence counter for every prefix of every suffix is incremented.

Example. Suppose the current leaf is $s_m = bde$ and the string de is one that crosses boundaries of existing s_i for $1 \leq i < m$. Thus de has not occurred as a prefix or a suffix of any s_i, $0 < i < m$. The set of proper suffixes of s_m is $S_m = \{\gamma, e, de\}$, and since there is no node representing the substring for de, it is created. Then the occurrence frequency is incremented for the first-level children b and d, and the new leaf node de. Figure 3.3 shows the tree after this transformation, which we call "LZ+prefix" or "LZP" for short.

We observe that this heuristic only discovers substrings that lie within a leaf string. Nonetheless, at this point it is possible to use the modified LZ tree and apply one of the existing prediction heuristics, e.g., use Eq. 3.2 and the Vitter-Krishnan method. Indeed, we include the LZP predictor in our comparison.

Similar to the LZ predictor, the LZP predictor maintains a LZP-tree T and the current tree node n, which corresponds to the current string s, that is, if we traverse the tree from node n up to the root of T, we get the inverse of the string s. Each node n has two properties: the location n_p and the value n_v. After a user arrives to a location p, it predicts the next location. We present the algorithm LZP-PREDICTION below.



$s = gbdcbgcefbdbde$

$s_i = \gamma, g, b, d, c, bg, e, ce, f, bd, de, bde$

Figure 3.3: Example LZP parsing tree

LZP-PREDICTION(n, s, p)

```
1  Q ← all children of node n
2  foreach node q in Q
3      do if q_p = p
4          then q_v ← q_v + 1            ▷ update the value of the matching node
5               n ← q                    ▷ move the current node to q
6               s ← s + p                ▷ update the corresponding string s
7               return FINDMAXCHILD(q)
8  add a new child q to node n
9  q_p ← p                               ▷ set the symbol of node q to p
10 q_v ← 1                               ▷ set the value of q to 1
11 s ← s + p                             ▷ update the corresponding string s
12 SUFFIX-UPDATE(s)                      ▷ Update the tree with all suffixes
13 n ← γ                                 ▷ set the current node to the root
14 return NIL
```

We show how to update the LZP-tree T using all suffixes of string s in SUFFIX-UPDATE.

```
SUFFIX-UPDATE(s)
 1  k ← LENGTH(s)
 2  if k = 0                                  ▷ check recursion stop condition
 3     then return
 4  q ← SUBSTR(s, 1, k − 1)
 5  n ← γ                                     ▷ the root of T
 6  while LENGTH(q) ≠ 0
 7        do o ← SUBSTR(q, 0, 1)
 8           if o matches a child c of n
 9              then c_v ← c_v + 1            ▷ update the value of the child node c
10              else
11                   add a new child c of n
12                   c_p ← o                  ▷ set the symbol
13                   c_v ← 1                  ▷ set the value
14           m ← LENGTH(q)
15           q ← SUBSTR(s, 1, m − 1)
16           n ← c                            ▷ set n to be its child
17  s ← SUBSTR(s, 1, k − 1)
18  SUFFIX-UPDATE(s)    update recursively
```

Bhattacharya and Das [BD02] propose a further heuristic that uses the LZP tree for prediction. This second heuristic is based on the Prediction by Partial Match (PPM) algorithm for text compression [BCW90]. (The PPM algorithm essentially attempts to "blend" the predictions of several $O(k)$ Markov predictors using "escape" probability; we describe it briefly below.) Note here, we use the PPM algorithm to work on the LZP tree, which is different from the PPM tree. Given a leaf string s_m, construct its set of proper suffixes S_m and update the LZP tree as described above. Then, for each such suffix, the heuristic

considers the subtree rooted at the suffix and finds all the paths in this subtree originating from this subtree root. (Thus these paths would be of length l for $l = 1, 2, \ldots t$, where t is the depth of the suffix from the root to the node in the LZ tree.) The PPM algorithm is then applied to this set of paths. PPM first computes the predicted probability of each path, and then uses these probabilities to compute the most probable next symbol(s) based on their weights (number of occurrences) in the path. We include the "LZ+prefix+PPM" predictor, nicknamed "LeZi" [BD02], in our comparison.

As in the other LZ-family predictors, the LeZi predictor LEZI-PREDICTION takes the LZP-tree T, the current node n, and the current new location p to predict the next location.

LEZI-PREDICTION(n, s, p)

```
1  Q ← all children of node n
2  foreach node q in Q
3      do if q_p = p
4          then q_v ← q_v + 1            ▷ update the value of the matching node
5               n ← q                    ▷ move the current node to q
6               s ← s + p                ▷ update the corresponding string s
7               return FINDMAXPPM(s)
8  add a new child q to node n
9  q_p ← p                               ▷ set the symbol of node q to p
10 q_v ← 1                               ▷ set the value of q to 1
11 s ← s + p                             ▷ update the corresponding string s
12 SUFFIX-UPDATE(s)                      ▷ Update the tree with all suffixes
13 n ← γ                                 ▷ set the current node to the root
14 return NIL
```

Finding the location of the child of node n with maximum probability from LZP tree T:

FINDMAXPPM(s)

1 $Q \leftarrow$ NIL ▷ list of (symbol, probability) pairs
2 $w \leftarrow 1.0$ ▷ escape probability
3 $n \leftarrow$ LENGTH(s)
4 **for** $i \leftarrow 0$ **to** n
5 **do** $l \leftarrow$ SUBSTR($s, i, n - i$) ▷ prefix string
6 $w \leftarrow$ PPM-PROBABILITY(Q, l, w)
7 **return** FINDMAX(Q)

We list the PPM algorithm used by FINDMAXPPM below.

PPM-PROBABILITY(Q, s, w)

```
1  k ← LENGTH(s)
2  n ← γ                                  ▷ the root of the tree
3  for i ← 0 to k − 1                     ▷ walk to the node corresponding to s
4      do o ← SUBSTR(s, i, 1)
5          if o matches a child c of n
6              then n ← c
7              else return 0.0            ▷ no corresponding node
8  m ← n_v
9  e ← m
10 Y ← all children of node n
11 foreach node y in Y                    ▷ sum up the node and all its children's values
12     do m ← m + y_v
13 foreach node y in Y
14     do u ← w ∗ y_v/m                   ▷ the probability of y with escape probability
15         if find (y_p, v) in Q
16             then v = v + u
17             else v = u
18         put (y_p, v) into Q            ▷ update the pair in Q
19 if k = 0
20    then return 0.0                     ▷ Q is built
21    else return w ∗ e/m                 ▷ return the escape probability
```

3.2.5 PPM predictors

Prediction by Partial Matching (PPM) is a data compression scheme, often used in text compression [CW84]. As with Markov predictors, the basic idea of PPM is to use the last few symbols in the input stream to predict the next one. Order-k (or $O(k)$) PPM uses the k immediately preceding symbols to update the model for prediction. An $O(k)$ PPM predictor blends a set of different-order context models, from 0 to k. Each fixed-order context model is built as an $O(k)$ Markov model. The combination of the different models is achieved through the use of "escape" probabilities, which are the probabilities of encountering previously unseen symbols. There are three major methods to determine the escape probabilities [BCW90]. The method used in our implementation is called "Method C" in [BCW90]. The number of escape events is equal to the number of different symbols that have been seen in the context so far. The total number of events is the number of all symbols that have been seen in the context so far. Thus the escape probability is the number of escape events divided by the total number of events. Let E_k be the escape probability,

$$E_k = \frac{N_e}{N}, \tag{3.3}$$

where N_e is the number of escape events, N is the total number of all events.

Then for all $a \in \mathcal{A}$, the $O(k)$ PPM probability is

$$P(x_{n+1} = a \mid L) = P_k(a) + \sum_{i=1}^{k} P_{i-1}(a) E_i, \tag{3.4}$$

where $P_k(a)$ is the probability computed using the $O(k)$ Markov model (see Equation 3.1).

The $O(k)$ PPM predictor maintains a full height-k tree. It takes the current context string s and the new location p, then predicts the next location.

PPM-PREDICTION(s, p, k)

1 $s \leftarrow s + p$ ▷ append p to s

2 SUFFIX-UPDATE(s) ▷ Update the tree with all suffixes

3 $s \leftarrow$ SUBSTR$(s, 1, k)$ ▷ update the current sequence s

4 **return** FINDMAXPPM(s)

3.2.6 SPM predictors

Jacquet, Szpankowski and Apostol [JSA00] proposed a predictor called the *Sampled Pattern Matching* (SPM) algorithm. This algorithm is of interest as it considers much larger classes of sources (in our case, handoff traces) than $O(k)$ Markov predictors. In particular, it addresses what are called strongly mixing sources, which contain fixed-order Markov sources as a special case. SPM was shown theoretically to be asymptotically optimal, that is, to have a fault rate the same as the best possible predictor, for this class of sources. SPM is similar to $O(k)$ Markov except instead of using a fixed value of context length k, the length is determined by a fixed fraction (α) of the longest context that has been previously seen, as described below.

Start by finding the longest suffix of $X_1, X_2, \ldots, X_t$ that has occurred previously in the sequence; this is called the maximal suffix, and t is the length of the suffix. Instead of considering the entire maximal suffix, however, the algorithm considers only a fraction of the suffix. Choose a fixed constant α, where $0 < \alpha < 1$. The suffix of interest is now $c = X_{t-k+1}, X_{t-k+2}, \ldots, X_t$ where $k = \lceil \alpha t \rceil$. The next predicted character is

$$\text{argmax}_{a \in A} N(ca, L),$$

where $N(s, L)$ is the number of times the string s occurred in the history L.

Example: Consider the sequence of length 40 below. The maximal suffix is YGSJSLJZ; if

we set $\alpha = 0.5$ then the fractional suffix is SLJZ. Its occurrences are shown below marked by a box:

$$\boxed{\text{SLJZ}}\text{GGDLYGSJ}\boxed{\text{SLJZ}}\text{KGS}\boxed{\text{SLJZ}}\text{ID}\boxed{\text{SLJZ}}\text{GGZYGSJ}\boxed{\text{SLJZ}}.$$

The symbols that followed the fractional suffix SLJZ in the sequence are G, K, I, G, respectively. SPM will predict G.

The SPM predictor takes the full location history L, and the current location p to predict the next location. We list the procedure below.

SPM-PREDICTION(p)

```
1  L ← L + p                                  ▷ append p to the history L
2  s ← FINDMAXSUFFIX(L)                       ▷ find the max suffix of the history
3  if s = NIL
4     then return NIL                         ▷ no prediction
5  t ← LENGTH(s)
6  m ← ⌈αt⌉
7  s ← SUBSTR(s, t − m, m)                    ▷ get the fractional suffix
8  n ← LENGTH(L)
9  Q ← NIL
10 for i ← 0 to n − m − 1
11       do l ← SUBSTR(L, i, m)
12          if l = s                          ▷ find the occurrence of s
13            then o ← SUBSTR(L, i + m, 1)
14                 if find (o, v) in Q        ▷ count the symbols following s
15                   then v = v + 1
16                   else  v = 1
17                 put (o, v) into Q
18 return FINDMAX(Q)
```

We use the brute force method to find the max suffix of the history. We examine the length-k suffix. If it exists in the previous history, we examine the length-$(k+1)$ suffix, for $k = 1, 2, \ldots, n-1$. We repeat this process until we found no length-k suffix in the previous history, then we return the length-$(k-1)$ suffix as the longest suffix.

FINDMAXSUFFIX(L)

1 $n \leftarrow$ LENGTH(L)

2 $a \leftarrow$ SUBSTR($L, 0, n-1$) ▷ history except the last symbol

3 **for** $k \leftarrow 1$ **to** $n-1$

4 **do** $b \leftarrow$ SUBSTR($L, n-k, k$) ▷ last k symbols of L

5 **if** b is not a substring of a

6 **then return** SUBSTR($b, 1, k-1$)

7 **return** NIL

3.2.7 CDF predictor

Rather than attempting to predict the next symbol or value in a sequence, this predictor works only with values and makes prediction about inequalities. Specifically, it produces the probability that the next value is less than (or greater than) a given value. It does so by computing the observed cumulative distribution function (CDF) of the historic values, and using the CDF to measure the probability of a given value appearing in the distribution.

Consider a history H of values $v_1, v_2, \ldots, v_n$. Suppose V is the random variable, which outputs the actual values in H, and P is its distribution. The CDF predictor computes the observed CDF function of V from the histogram, that is,

$$\text{CDF}(V < v) = \frac{1}{n} \sum_{i=1}^{n} I(v_i < v), \tag{3.5}$$

where I is the indicator function. In a similar fashion, we can compute the probability of values occurring in range $a \leq V < b$, by simply computing

$$P(a \leq V < b) = (\text{CDF}(V < b) - \text{CDF}(V < a)). \tag{3.6}$$

We present our CDF duration predictor's algorithm as a procedure CDF-PREDICTION that takes a sequence of durations H, and the value v. It computes the probability of that next duration is less than v.

CDF-PREDICTION(H, v)

1 $n \leftarrow$ LENGTH(H)

2 $c \leftarrow 0$

3 **foreach** v_i in H

4 **do if** $v_i < v$

5 **then** $c \leftarrow c + 1$

6 **return** c/n ▷ the probability

3.2.8 Moving average predictors

Moving averages are commonly used to predict a trend in a sequence of values. The order-k (or "$O(k)$") *moving-average* (MA) predictor takes a sequence of values and predicts that the next value of the sequence is the average of the last k values in the sequence.

Consider a history H of values $v_1, v_2, \ldots, v_n$. The order-k moving-average predictor estimates the next value using the following equation. Let $k' = \min\{k, n\}$, then

$$v_{n+1} = \frac{1}{k'} \sum_{i=1}^{k'} v_{n-i+1}. \tag{3.7}$$

We present the moving-average predictor as follows.

MA-PREDICTION(H, k)

```
1  n ← LENGTH(H)
2  c ← 0
3  if k > n
4     then i ← n − k + 1
5     else  i ← 1
6           k ← n
7  while i ≤ n
8        do c ← c + v_i
9           i ← i + 1
10 return c/k                          ▷ the average duration
```

3.2.9 Combining location and time prediction

Many applications need to predict both the time and destination of the next handoff. The goal is to predict, for every possible destination, the probability that a handoff will occur within the next Δt period, conditioned on the current location and the duration of the visit so far. Our approach is to combine the Markov location predictor and the CDF time predictor.

Consider a user's handoff history $H = (t_1, a_1), (t_2, a_2), \ldots, (t_n, a_n)$, in which t_i is the time that the user arrived at location a_i. From H we extract the location history $L = a_1, a_2, \ldots, a_n$, and from L the order-k location context $c = L(n - k + 1, n) = a_{n-k+1}, \ldots, a_{n-1}, a_n$. We then search the history L for instances of the context c. We need to examine the destinations that follow each such instance, and in particular to examine the duration of the visit preceding each of those destinations, to be able to predict the duration in the current context. So, we need to extract the set of durations for each possible

destination x:

$$D_x = \{d_i \mid d_i = t_{i+1} - t_i \text{ where } L(i-k+1, i+1) = cx\} \tag{3.8}$$

.

From each duration set D_x, we use the CDF predictor to compute the conditional probability $P_x(0 \leq d < d_0 + \Delta t \mid c, d_0)$ that the user will move to location x within Δt seconds after the user has stayed d_0 seconds. We can also use the $O(k)$ Markov predictor to compute the probability $P(x)$ of every possible next location x with Equation (3.1).

Therefore, the probability of the user moving to each of the possible locations x within the next Δt seconds after spending d_0 seconds at the current location, given the current context c.

$$P(x, d_0 \mid c) = P(x) \cdot P_x(0 \leq d < d_0 + \Delta t \mid c, d_0) \tag{3.9}$$

We name this predictor MarkovCDF.

This predictor is always conditioned on the current location context. As described above, the predictor builds per-user tables, but it is equally possible to build aggregate tables from all users' handoff histories. In the latter case, all users share the same Markov transition matrix, and update the matrix whenever they move.

We present the predictor's procedure MARKOVCDF-PREDICTION as follows. The procedure takes the Markov matrix M, duration histories at each location $H = H_i, H_j, \ldots$, and the context string c. The user moved to the current location at t_0. At time t, the predictor computes a list of location and probability pairs; the user may move to each of the locations with the corresponding probability within duration d.

MARKOVCDF-PREDICTION(M, H, c, t_0, t, d)

1 $Q \leftarrow M_c$ ▷ symbols from the row c of M
2 $X \leftarrow$ NIL
3 $m \leftarrow 0$
4 **foreach** pair (p, v) in Q ▷ sum of all values
5 **do** $m \leftarrow m + v$
6 **foreach** pair (p, v) in Q
7 **do** $H_p \leftarrow$ the duration of H at p
8 $u \leftarrow$ CDF-PREDICTION$(H_p, t - t_0 + d) -$ CDF-PREDICTION$(H_p, t - t_0)$
9 $u \leftarrow uv/m$
10 add pair (p, u) into X
11 **return** X

Note that this predictor is unlike the others above in that it returns a set of probable destinations, rather than the most probable destination.

3.2.10 Neighbor Graph predictor

Using users' current neighbor locations as the prediction is an obvious way to predict future locations. The predictor is called *Neighbor Graph* (NG) predictor. The NG predictor outputs a vector of probabilities, one for each AP that is the neighbor of the current AP. The probabilities depend only on the locations that the user visited in the history. Mishra et al. [MSA04] present an algorithm to dynamically build a user's neighbor graph to cache context for fast handoffs. In fact, the NG predictor behaves just like the $O(1)$ Markov predictor except it outputs a list of location and probability pairs, but the $O(1)$ Markov handoff-location predictor predicts only the most likely location. We compared the performance of our predictors to this simple predictor in the studies of over-provisioning and

under-provisioning (Section 4.2.3), and bandwidth-reservation schemes of VoIP applications (Section 5.4).

3.3 Prediction metrics

To evaluate the performance of predictors, we define a set of metrics that measures the performance of location prediction, time prediction, and joint time and location prediction.

3.3.1 Next-location prediction

During an on-line scan of the location history, the predictor is given a chance to predict each location. There are three possible outcomes for this prediction, when compared to the actual location:

- The predictor correctly identified the next location.
- The predictor incorrectly identified the next location.
- The predictor returned "no prediction."

All predictors encounter situations in which they are unable to make a prediction; in particular, all history-based predictors will have no prediction for the first location of each user trace.

We define the *accuracy* of a predictor for a particular user to be the fraction of moves for which the predictor correctly identified the next location. Thus, "no prediction" is counted as an incorrect prediction. In the future we plan to examine other metrics that can better distinguish the two forms of incorrect prediction (there may be some applications that may prefer no prediction to a wild guess). For now, this metric best reflects the design of predictors common in the literature.

If the location has geographical coordinates, other accuracy metrics could be measured. For example, the distance between the actual location and the predicted location, or the angle between the direction to the actual location and the direction to the predicted location, are two non-binary accuracy metrics. Since our data are symbolic, these geographic metrics do not apply to our predictors.

3.3.2 Time prediction

Time is a continuous value. It is impossible to predict the exact time of an event, so we must predict the handoff time as a period. We can also quantize the actual handoff time using this period and compare them. The period can be seconds, minutes, or even hours. Then we can again use the ratio of the number of correct predictions to the number of all handoffs to define accuracy. However, this definition may be misleading, because large periods are inherently easy to predict. One can use an arbitrary large period to get 100% accuracy.

Instead, we measure how much the predicted handoff time differs from the actual handoff time. In most applications, overshooting or undershooting the actual time of handoff will have different implications; for example, a late request to reserve bandwidth for a roaming voice call will be useless, whereas an overly early request will consume excess resources.

Hence, we define separate metrics; when a handoff occurs at time t_h, and the handoff was predicted to occur at time t_p, the prediction *earliness* is $t_h - t_p$ if $t_h \geq t_p$, and *lateness* is $t_p - t_h$ if $t_h < t_p$. We compute each user's average earliness across all handoffs in which the prediction was early, and each user's average lateness across all handoffs in which the prediction was late.

3.3.3 Joint time and location prediction

We use over-provision and under-provision to measure the joint time and location prediction. Over-provision and under-provision are metrics that relate to the application of resource allocation. When a predictor is used for advance resource allocation, an incorrect prediction can lead the system to reserve too many, or too few, resources. We consider predictors, such as the CDF predictors defined in the preceding section, that provide the probability of handoff for each possible destination, and imagine a system that provisions resources at each destination in proportion to these probabilities. When the handoff occurs, we can determine the amount that resources were under-provisioned at the actual destination and over-provisioned at all other locations.

Under-provisioning: At any time t that the predictor indicates that the user u may move from location i to location j with probability $P(j \mid u, i, t)$, and the handoff occurs, we measure the amount of under-provisioning for this handoff as

$$E(u, i, j, t) = 1 - P(j \mid u, i, t).$$

In effect, under-provisioning is equivalent to computing the error with respect to an "ideal predictor" that always returns the correct prediction: probability 1 for the correct destination just before the handoff is to occur, and probability 0 for other destinations or other times. Thus, a lower value of E represents better prediction quality. When a predictor is used periodically, as might the CDF predictors defined in the preceding section, under-provisioning applies only to the last prediction before the handoff.

Over-provisioning: Any time t that the predictor indicates that the user u may move from location i to location j with probability $P(j \mid u, i, t)$, and a system provisions resources in accordance with that probability, the resources are wasted if the handoff does

not occur as predicted. We define the amount of over-provision as

$$F(u, i, j, t) = P(j \mid u, i, t),$$

because an ideal predictor would have produced probability zero. Thus, a lower value of F represents better prediction quality. When a predictor is used periodically, we can compute the over-provision metric for each such prediction, under the rationale that a system using such a predictor would waste resources at predicted destinations whenever a handoff does not occur, or at all locations other than the actual destination whenever a handoff does occur, within that prediction interval.

As a more comprehensive representation of these metrics, we can compute per-user and per-location averages; a non-uniform distribution would mean that some users or some locations were more amenable to prediction than the rest. We define the average under-provision per user as

$$E(u) = \frac{\sum_{i,j,t} E(u, i, j, t)}{\sum_{i,j,t} 1},$$

and the average under-provision per access point as

$$E(j) = \frac{\sum_{u,i,t} E(u, i, j, t)}{\sum_{u,i,t} 1}.$$

In the last three sections, we describe the predictors that we evaluated, the metrics that we used, and the predictors' uses. We show the evaluation results of the predictors according to these uses in the next chapter.

3.4 Correlation of prediction accuracy and entropy

We need one more set of definitions before we can explore the results in the next chapter. We expect that some correlation exists between prediction accuracy and the entropy of user's handoffs. Before we show the simulation results, we first define the entropy of users' handoffs.

3.4.1 Definition of the Entropy

The standard definitions relating to entropy are as follows:

Definition 1: The entropy $H(X)$ of a discrete random variable X is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} P(x) \log P(x),$$

where $\mathcal{X}$ is the set of all possible values of X, and $P(x)$ is the probability of $X = x$.

The entropy is used to describe the uncertainty of a random variable. In other words, how much information on average is needed if we want to determine the value of X. The more uncertain a random variable is, the higher the entropy.

Definition 2: The joint entropy H(X,Y) of a pair of discrete random variables (X,Y) is defined as

$$H(X,Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x,y) \log P(x,y),$$

where $P(x,y)$ is the probability of the pair (x,y).

Similarly, the joint entropy describes how much information on average is needed if we want to determine both values of X and Y.

Definition 3: The conditional entropy $H(X \mid Y)$ is defined as

$$H(X \mid Y) = -\sum_{y \in \mathcal{Y}} P(y) \sum_{x \in \mathcal{X}} P(x \mid y) \log P(x \mid y),$$

Conditional entropy describes how much more information on average is needed if we want to determine the value of X when the value of Y is known.

3.4.2 Entropy of an *O(k)* Markov predictor

For an $O(k)$ Markov predictor, the prediction conditions on the most recent k symbols. The probability that the next symbol is a is

$$P_k(a) = \widehat{P}(X_{n+1} = a \mid L) = \widehat{P}(X_{n+1} = a \mid c) = \frac{N(ca, L)}{N(c, L)},$$

where c is the most recent k symbols, L is all the sequence of symbols in the history, and $N(s', s)$ denotes the number of times the substring s' occurs in the string s. The predictor always chooses $X = a$ that leads to maximum $\widehat{P}_k(X)$ as the prediction.

The conditional entropy is then calculated as

$$H(A \mid C) = -\sum_{c \in \mathcal{C}} P(c) \sum_{a \in \mathcal{A}} P(a \mid c) \log P(a \mid c), \tag{3.10}$$

where A is a discrete variable choosing from the all possible symbols in set $\mathcal{A}$, and C is a discrete variable choosing from the possible k-symbol context set $\mathcal{C}$. $P(c)$ is the estimation of the probability of a given k-symbol context in the whole history:

$$P(c) = \frac{N(c, L)}{\sum_{y \in \mathcal{C}} N(y, L)}.$$

3.4.3 Example

In our prediction research, the input to a predictor is a random variable. Entropy is used to describe the characteristic of a random variable. As a result, it is independent of the prediction algorithm. The conditional entropy, however, depends on the "condition" that we use. As a result, we have different conditional entropy calculations for different predictors that use different length contexts.

For example, consider a history $L = abacaba$ from an alphabet $\mathcal{A} = \{a, b, c\}$.

For an $O(1)$ Markov predictor, the length-1 context variable, $\mathcal{C}$, has 3 possible values, $\{a, b, c\}$, and has 6 length-1 context occurrences (the last symbol is not counted, because we do not know the following symbol yet). We get $P(C = a) = 3/6$, $P(C = b) = 2/6$, and $P(C = c) = 1/6$ in this example. We also get $P(b \mid a) = 2/3$, $P(c \mid a) = 1/3$, $P(a \mid b) = 1$, and $P(a \mid c) = 1$. To compute the entropy, we use the log base to be 2 for simplicity. Because we care only about the relative entropy, which log base we use does not matter in our correlation study. The conditional entropy considering length-1 context for this example is $H(A \mid C) = [\frac{3}{6}(\frac{2}{3}\log\frac{3}{2} + \frac{1}{3}\log 3) + \frac{2}{6}(1\log 1) + \frac{1}{6}(1\log 1) \approx 0.46]$.

For an $O(2)$ Markov predictor however, the length-2 context set is $\mathcal{C} = \{ab, ba, ac, ca\}$, and has 5 length-2 context occurrences (the last length-2 context is not counted, because we do not know the following symbol yet). We get $P(C = ab) = 2/5$, $P(C = ba) = 1/5$, $P(C = ac) = 1/5$, and $P(C = ca) = 1/5$. We also get the conditional probabilities: $P(a \mid ab) = 1$, $P(c \mid ba) = 1$, $P(a \mid ac) = 1$, and $P(b \mid ca) = 1$. The conditional entropy, considering the length-2 context in this example, is $H(A \mid C) = [\frac{2}{5}(1\log 1) + \frac{1}{5}(1\log 1) + \frac{1}{5}(1\log 1) + \frac{1}{5}(1\log 1) = 0]$.

3.5 Related work

Many handoff prediction techniques have been proposed in many areas of wireless cellular networks as a means of enhancing performance, including better mobility management [LM96], improved assignment of cells to location areas [DS99], more efficient paging [BD02, KT04], call admission control [YL02], and location-based services [KL03]. Cheng, Jain and van den Berg surveyed a set of location-prediction techniques [CJv03]. Relatively less prediction study has specifically been done for WLANs. Here we just briefly survey some of related work on handoff prediction.

3.5.1 Prediction for fast handoff

Mishra et al. [MSA04] propose to push the "context" of a mobile device from one AP to another using the inter-access point protocol (IAPP). The context includes various types of information such as accounting and authentication information, IP flow information, QoS information, and header compression information. The goal was to reduce handoff latency and they achieved a factor of 10 improvement in their experiments. The solution was simple: each AP remembers the set of other APs from which clients roam, and thus builds an adjacency list representation of its portion of the "neighbor graph." They did not address the timing of the context push: it would appear that the APs push contexts as soon as the client arrives, which begs the following question: when should the context be pushed again for any kind of context that changes while the client remains at the AP? The authors did not make any effort to select a subset of neighbors to push the context: they assume that the number is small, and they push context to *all* neighbors observed recently. Similarly, Pack and Choi [PC04] propose a key-distribution algorithm that distributes keys to previously observed neighboring APs to reduce the authentication time.

Gown et al. [GKDJ05] show experimentally that IP header compression can be used to

reduce wireless bandwidth usage for Voice over IP (VoIP) applications running over Wi-Fi. In such a scenario, the system could place the context required for header compression at the next AP(s) where the user is predicted to move, to start header compression sooner and make the handoff smoother.

3.5.2 Bandwidth reservation

In wireless networks, the SC (Shadow Cluster) scheme can provide consistent QoS support before and after a handoff [LAN95]. In the SC scheme, all cells neighboring a mobile node's current cell are requested to reserve resources for that node, in case it moves to that cell next. Clearly, this approach ties up more resources than necessary. Several proposed SC variations [SG98] attempt to predict the device's handoff so the network can reserve resources in only certain neighboring cells.

Prihandoko et al. [PHA04] propose a prediction algorithm named UMP. Their UMPx model is the same as our order-k Markov with fallback model (Section 3.2.2), where $k = x - 1$. In this paper, the authors used the hexagon cell model to simulate a mobile environment. The call arrival rate was Poisson distributed and the call duration was exponentially distributed. The bandwidth required was in a geometrical distribution between the minimum and the maximum. The authors did not mention what mobility model they used in the simulation. They compared the simulation results of UMP5 and UMP2, and found that UMP5 was better in terms of handoff call-dropping probability. However, in terms of call blocking probability and bandwidth utilization, UMP5 produced similar results to UMP2. They did not compare other order models. In our results (Chapter 4), the order-2 Markov with fallback was the best, which would correspond to their UMP3 model. They did not compare UMP3 to the other models, however.

3.5.3 Other

Samaan and Karmouch [SK05] use a user's context, such as interests, activity, or tasks, to predict the user's next destination. A destination is a landmark object, e.g., computer science department or library. Then it uses a map to find the shortest path, as predicted trajectory, for the user to reach the destination. This paper assumes that location information is an existing context, and it assumes that knowledge about a user's schedule can be obtained through class registration, that tasks can be extracted from the user's electronic diary, and that interests can be collected through a questionnaire list. This prediction approach does not use historical mobility information of a user, thus avoiding the non-prediction problems when the user goes to a new location. Instead, this paper defines the predictability of a future location as the probability that their prediction algorithm believes the move. The total degree of predictability of a user's trajectory is the average of all visited location predictabilities. The authors derived prediction accuracy from the user's trajectory predictability. They measure the trajectory accuracy by determining the distance between predicted trajectory and the actual trajectory. Gathering more contextual information about users, other than location and handoff time, may not be easy, because of privacy concerns. It may only be possible for a group of controlled users, who may be asked to volunteer information.

In this chapter, we discuss predictors that use a history of handoffs between symbolic locations handoffs to predict a user's next location and to estimate the probability of handoff within a given time period. We also introduce the metrics that we used to evaluate these predictors and we define the entropy that we used to measure the correlation between predictor accuracy and the inherent predictability of a sequence. In the next chapter, we show our evaluation results.

Chapter 4

Predictor Evaluation

In this chapter we examine the simulation results of the handoff-location predictors, handoff-time predictors, and joint-time-and-location handoff predictor using our Wi-Fi handoff data.

4.1 Location Prediction

To drive our simulations, we used the traces that were collected between April 2001 and March 2003. During this period, there were a total of 6,202 users and 592 APs.

Before we discuss the details of the simulation results, we first discuss an inherent problem in prediction algorithms: how to handle ties.

4.1.1 Breaking ties

In our descriptions of the predictors (Chapter 3), we indicate that each predictor predicts the symbol with the highest probability of occurring next, given the current state of its data structure (Markov matrix or LZ tree, for example). It is quite possible, though, for there to be a tie for first place, that is, several symbols with equal probability and none with

higher probability. We have not found any discussion of this issue in the literature, yet our implementations must make a specific choice. We implemented three different tie-breaking methods:

- *First added:* among the symbols that tie, we predict the first one added to the data structure, that is, the first one seen in the location history.
- *Most recently added:* among the symbols that tie, we predict the one that was most recently added to the data structure.
- *Most recent:* among the symbols that tie, we predict the one most recently seen in the history.

In Figure 4.1, we show the ratio of the number of ties to the number of predictions, using a cumulative distribution function (CDF) across all users having more than 1000 movements for different predictors. Figure 4.1 shows that most of the users had few ties: ties occurred in fewer than 10% of all their individual predictions. Our experiments show that the results were not significantly affected by the choice of a tie-breaking method. Figure 4.2 shows the prediction accuracy for $O(1)$ Markov predictors with three different tie-breaking methods. The results of the three tie-breaking methods are almost indistinguishable. Therefore, we decided to use the "first added" method throughout the results below, and we did not consider tie-breakers any further.

4.1.2 Markov

Recall that accuracy is defined as the number of correct predictions divided by the number of attempts (moves). As we step through the locations in a single trace, attempting to predict each location in sequence, we can examine the accuracy up to that point in the trace. Figure 4.3 shows how the accuracy metric varies over the course of one user's history, using

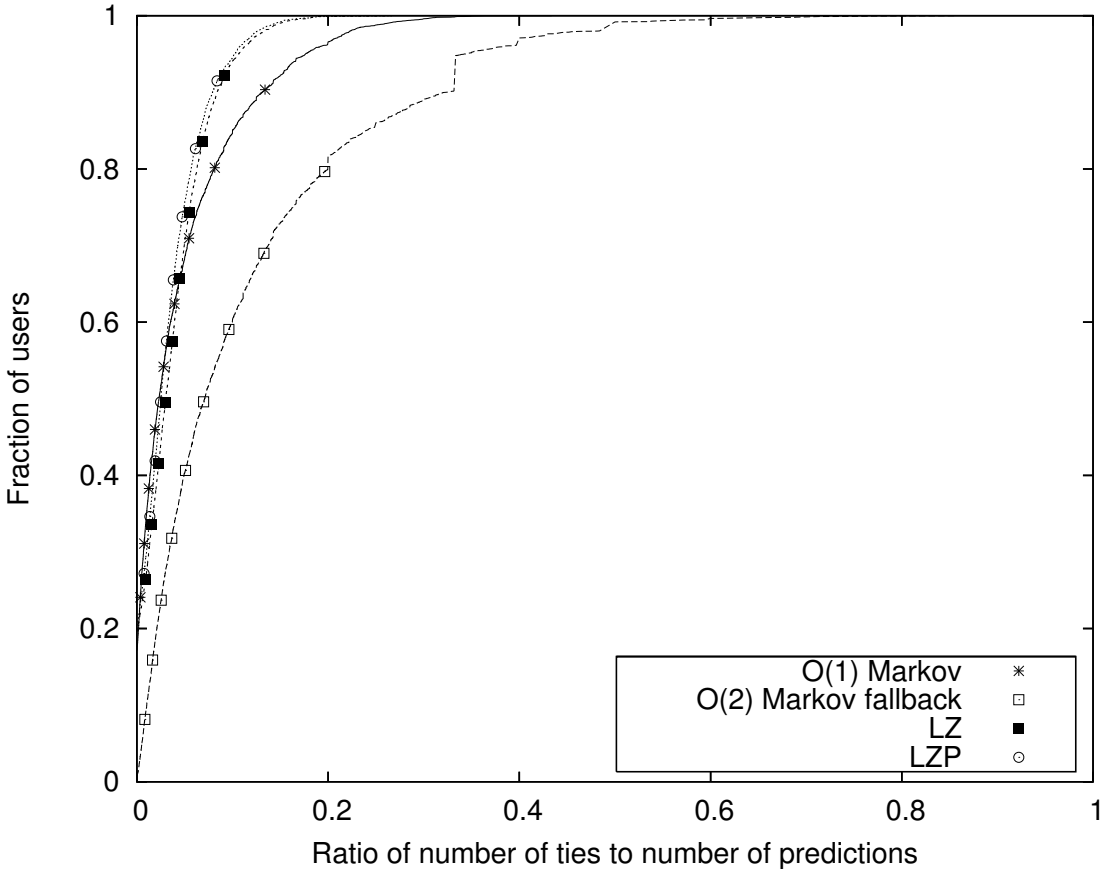


Figure 4.1: Ratio of number of ties to the number of predictions

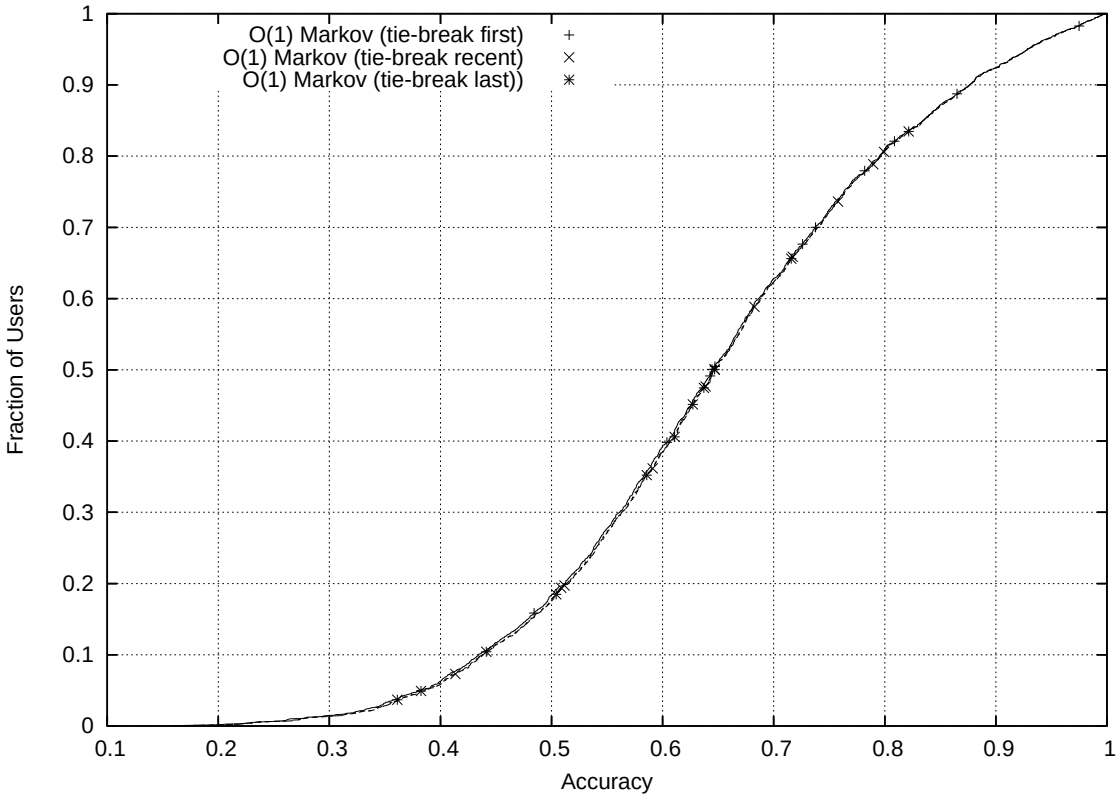


Figure 4.2: Compare different tie-breaking methods

the $O(1)$ Markov predictor. Ultimately, for comparison purposes, we record the accuracy over the entire trace, the rightmost value in this plot. In subsequent graphs we use this overall accuracy as the performance metric.

Of course, we have several thousand users, and the predictor was more successful on some traces than on others. In Figure 4.4 we display the accuracy of the $O(1)$ Markov predictor in CDF curves, one for each of three groups of traces: *short*, *medium*, and *long* traces, defined respectively as those with 100 or fewer moves, 101–1000 moves, and over 1000 moves. The predictor was clearly unsuccessful on most short traces, because its curve is far to the left. Ultimately, we found that all predictors fared very poorly on short traces and somewhat poorly on medium traces. For the sake of completeness, we generally consider all 6,202 traces. In some cases we do only consider long traces, when doing so makes trends clearer; in that case we note so explicitly.

Intuitively, it should help to use more context in each attempt to predict the next location. In Figure 4.5 we compare the accuracy of $O(1)$ Markov with that of $O(2)$, $O(3)$, and $O(4)$ Markov predictors. As an aside, we include an order-0 Markov predictor, in which no context is used in the prediction of each move. This predictor simply predicts the most frequent location seen so far in that user's history. Although it represents a degenerate form of Markov prediction, it helps to show the benefit of context in the $O(1)$ predictor.

Surprisingly, the $O(2)$ Markov predictor had similar median accuracy to the $O(1)$ Markov predictor, though with a wider spread in its accuracy. In fact, when considering only long traces (Figure 4.6), $O(2)$ generally outpredicted $O(1)$. The high-order $O(3)$ and $O(4)$ predictors were, however, worse than $O(2)$. Although these predictors use more information in the prediction process, they are also more likely to encounter a context (a three- or four-location string) that has not been seen before, in which case they are unable to make a prediction. A missing prediction is not a correct prediction, and these unpredicted moves bring down the accuracy of the higher-order predictors. In Figure 4.7 we display the

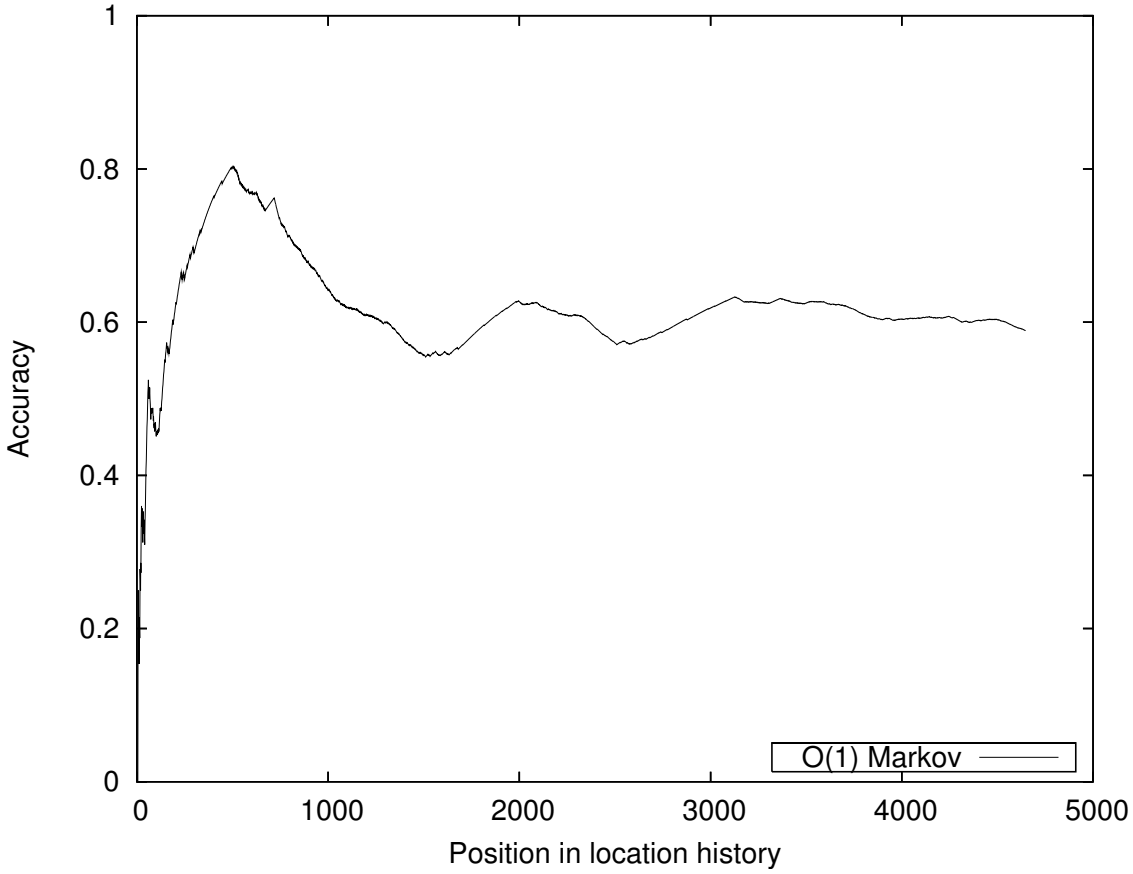


Figure 4.3: Prediction accuracy for a sample user

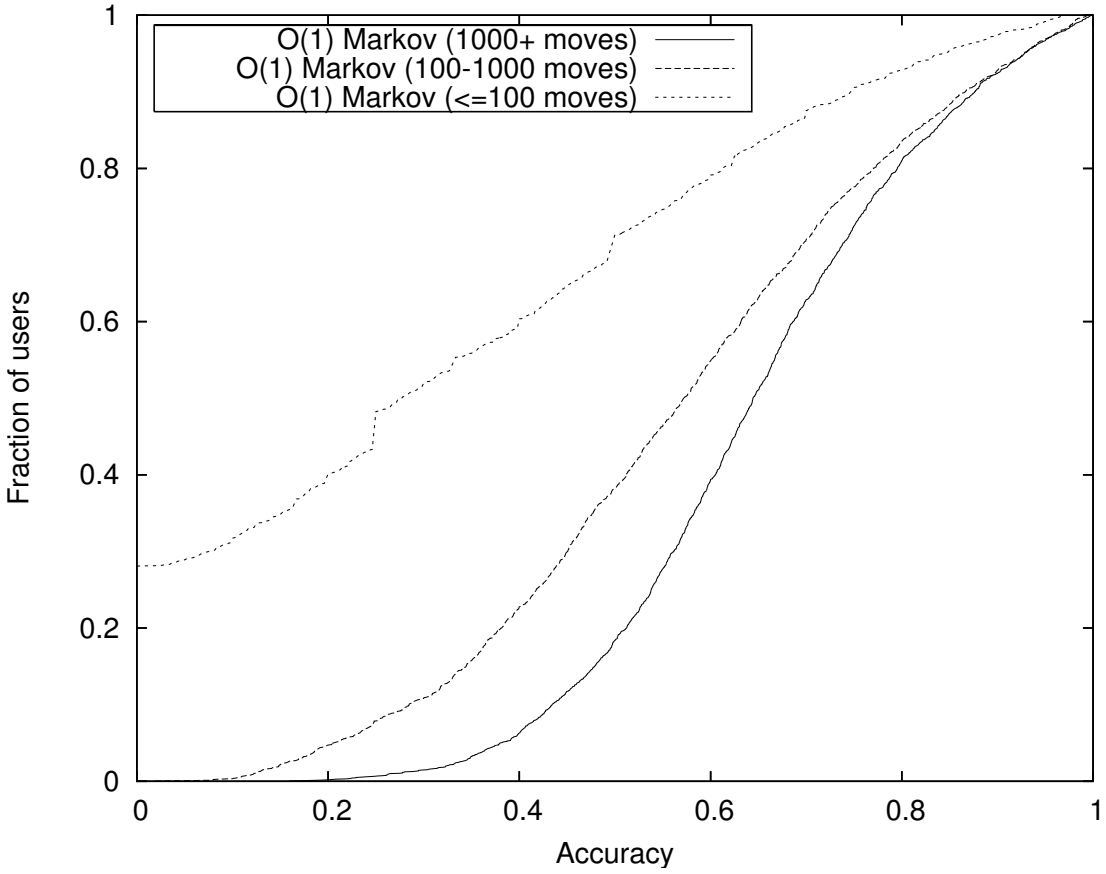


Figure 4.4: Accuracy of $O(1)$ Markov predictor

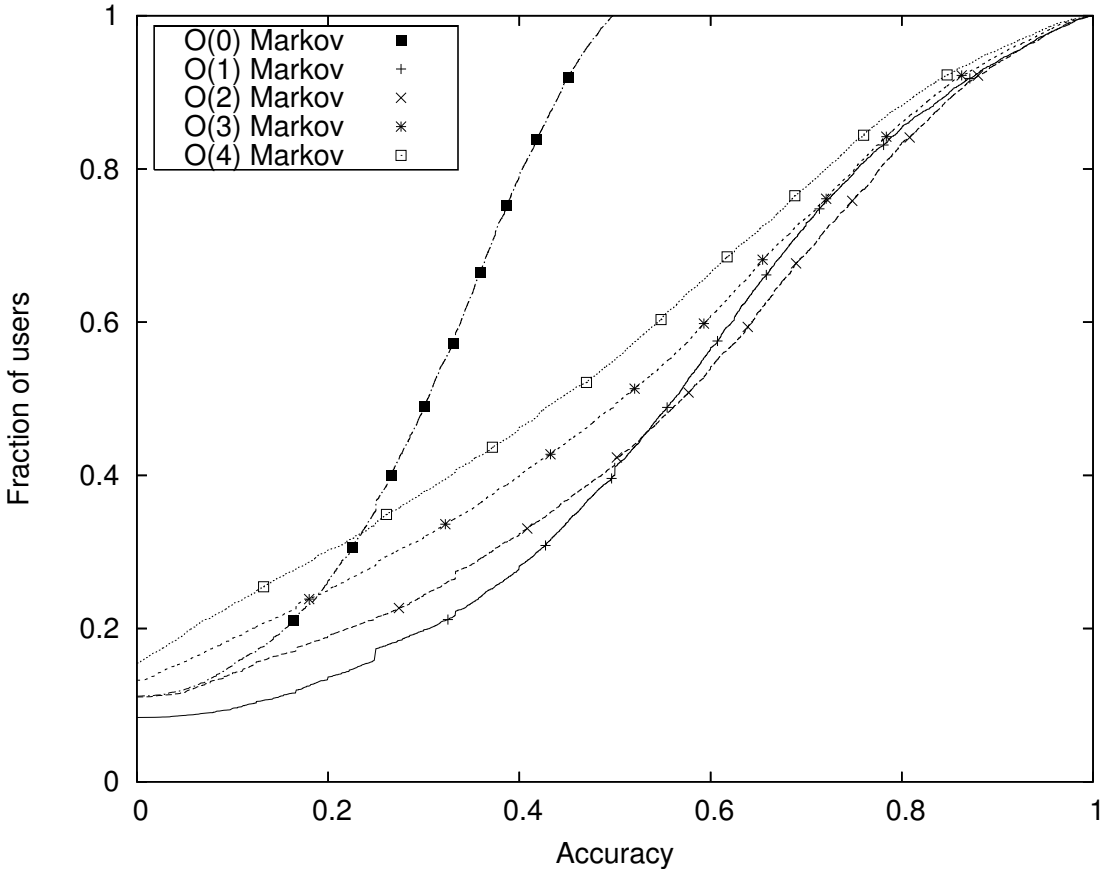


Figure 4.5: Comparing Markov predictors

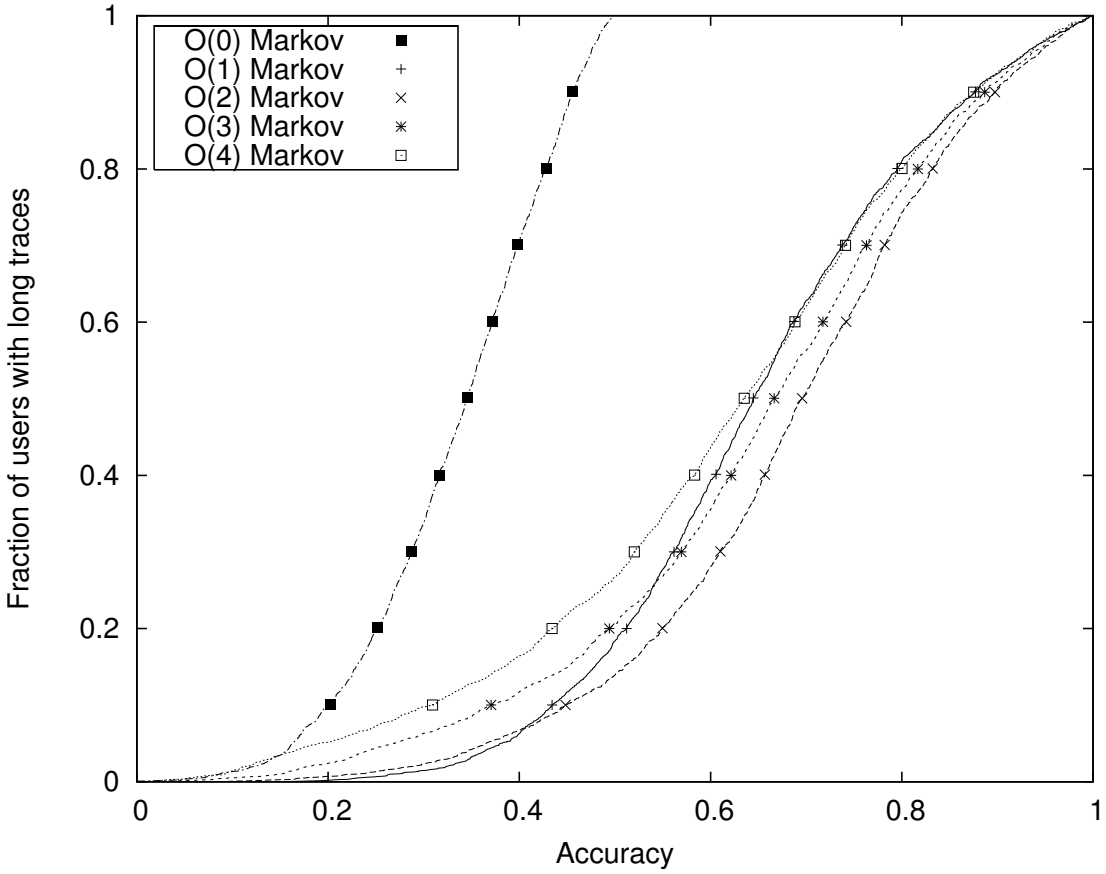


Figure 4.6: Comparing Markov predictors for long traces

conditional accuracy of these same predictors: the number of correct predictions divided by the number of predictions. In this metric, we ignore unpredicted moves, and it becomes clear that longer context strings did lead to better prediction, where possible, although with diminishing returns. Notice that, in Figure 4.7, 10–20% of the users had an accuracy of 0. The very short trace length is the reason for those 0 accuracies. This point becomes clear when we consider only long traces in Figure 4.8.

Returning to our original accuracy metric, we now consider our fallback mechanism for Markov predictors (Section 3.2.2). Figure 4.9 displays the performance of the $O(2)$ Markov predictor with fallback, which uses the results of the $O(2)$ predictor when it makes a prediction, or the $O(1)$ predictor if the $O(2)$ predictor has no prediction, or the $O(0)$ predictor if neither have a prediction. Fallback indeed helped to improve the $O(2)$ Markov predictor's performance. The $O(3)$ and $O(4)$ Markov predictor also improved with fallback, but the curve of $O(4)$ with fallback is nearly identical to the curve of $O(3)$ with fallback.

Markov predictors with fallback gain the advantage of the deeper context but without losing accuracy by failing to predict in unknown situations. The intuition is that a higher-order Markov predictor should perform better than a lower-order one. In fact, this is not always correct. A higher-order Markov predictor uses more context information, but it has fewer samples than a lower-order one with the same movement history. For example, in our experiments, we found that the $O(3)$ Markov with fallback predictor performed worse than the $O(2)$ Markov with fallback predictor (Figure 4.9). In the cases where the $O(3)$ predictor made an incorrect prediction but the $O(2)$ did not, we found that the $O(3)$ Markov predictor had an average of 8.23 samples, with a median number of 1.96, while the $O(2)$ Markov predictor had an average of 45.5 samples, with a median number of 12.2.

In the plots so far, we have examined the performance of $O(k)$ Markov predictors that used the most recent k locations in making a prediction, weighting the potential next loca-

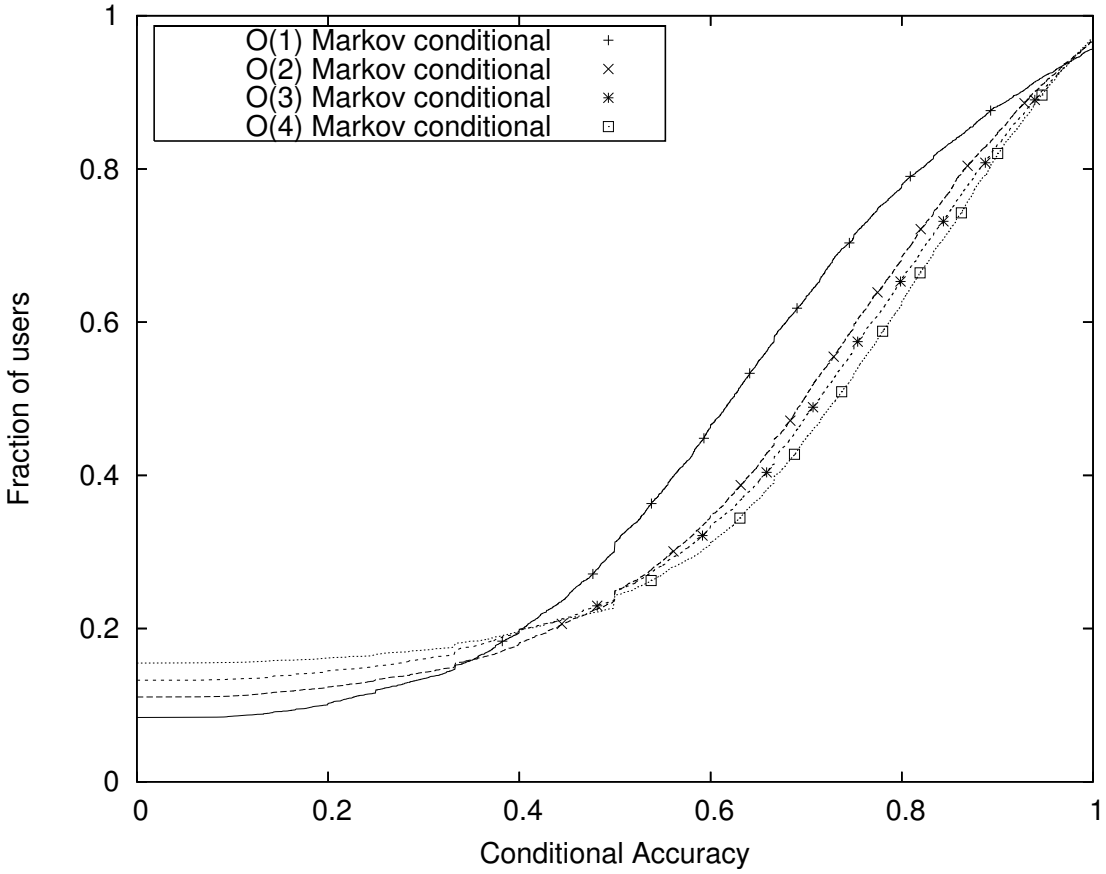


Figure 4.7: Conditional accuracy metric for all traces

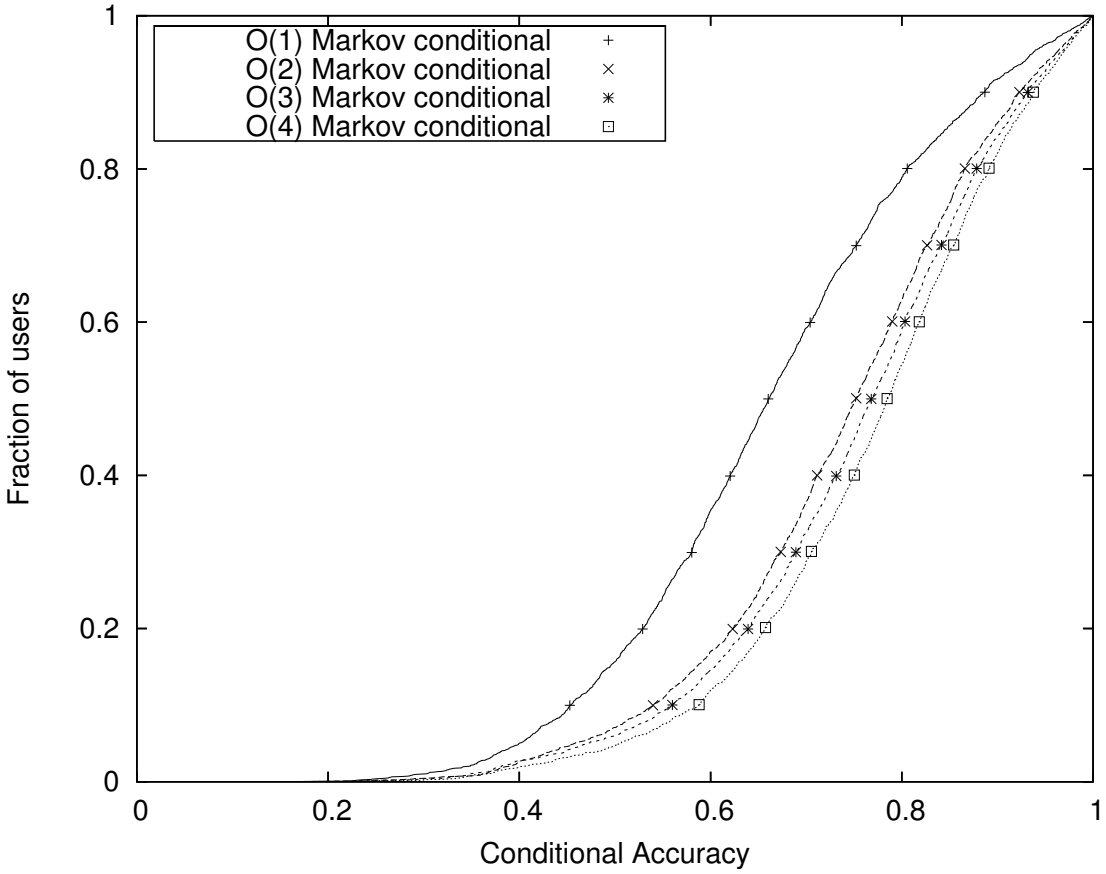


Figure 4.8: Conditional accuracy metric for long traces

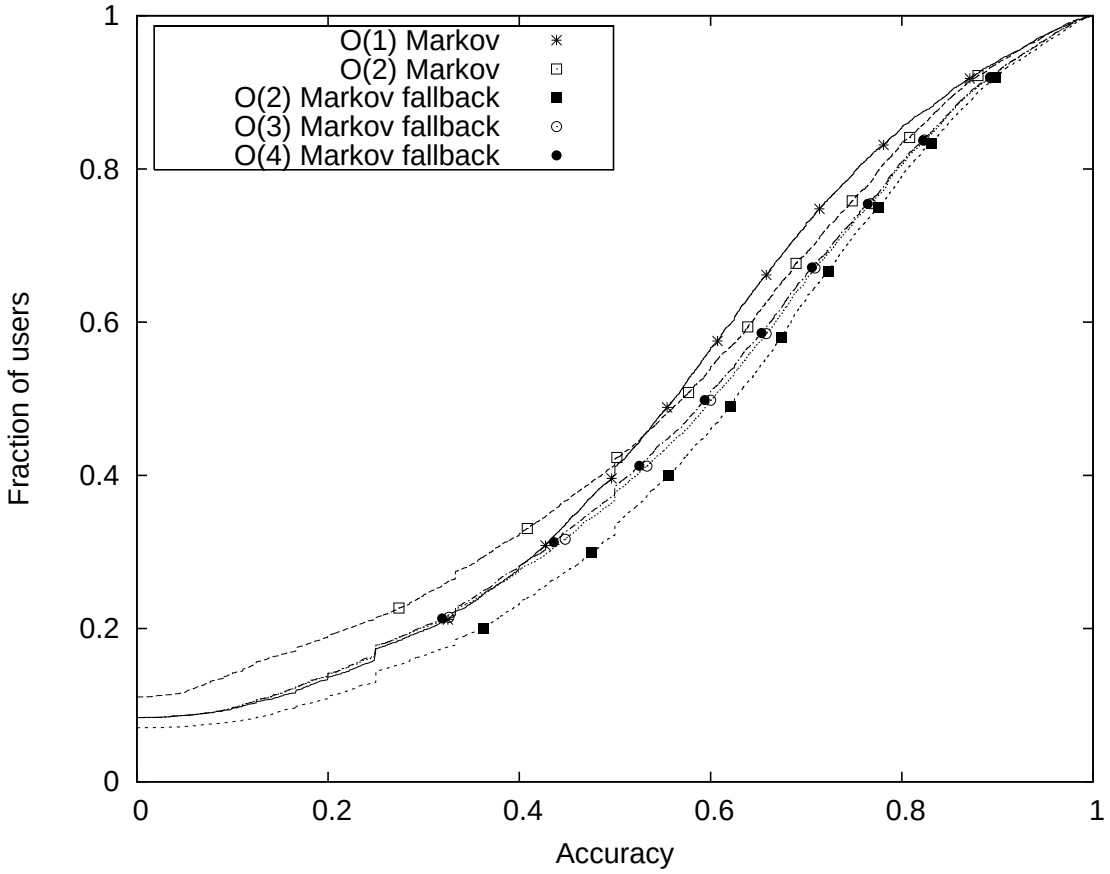


Figure 4.9: Markov predictors with fallback

tions by their frequency of occurrence in the past. An alternative approach assigns weights according to the recency: how recently the location occurred. Thus, one transition (the most recent seen for this context) has weight 1 and the others have weight 0. We can keep only the most recent transition of each context, instead of all possible transitions. Observe that the $O(k)$ Markov prediction using the most frequent location in general takes order n^{k+1} space for an n-location environment, while using the most recent location it takes only order n^k space, a potentially significant decrease in storage. Figure 4.10 shows the quality of Markov predictors based on this approach. Curiously, here $O(2)$ did worse than $O(1)$, although fallback made up some of the difference.

In Figure 4.11 we compare the recency-weighted approach with the original frequency-weighted approach. The best recency-weighted Markov predictor, $O(1)$, was better than the corresponding frequency-weighted predictor. This result implies that recent history was a better predictor of the immediate future than was an accumulated probability model, when considering only the current location. On the other hand, recall from Figure 4.10 that among the recency-weighted predictors the extra context of $O(2)$ lowered prediction accuracy. Thus, among $O(2)$ predictors, the frequency-weighted approach beats the recency-weighted approach (not shown in Figure 4.11). Ultimately, the $O(2)$ frequency-weighted Markov predictor with fallback had the best outcome.

Some of our user traces span a period of hours or days, but some span weeks or months. Clearly it is possible for a user's handoff patterns to change over time; for example, a student may move to a new dormitory, or attend different classes. The transition weights constructed by the frequency-weighted Markov model may become ineffective after such a change; for users with a long history, these weights will adapt too slowly. In another series of experiments (Figure 4.12), we added an exponential decay to the frequency weights so that more recent locations have a larger impact, but we saw little improvement in the quality of the predictors.

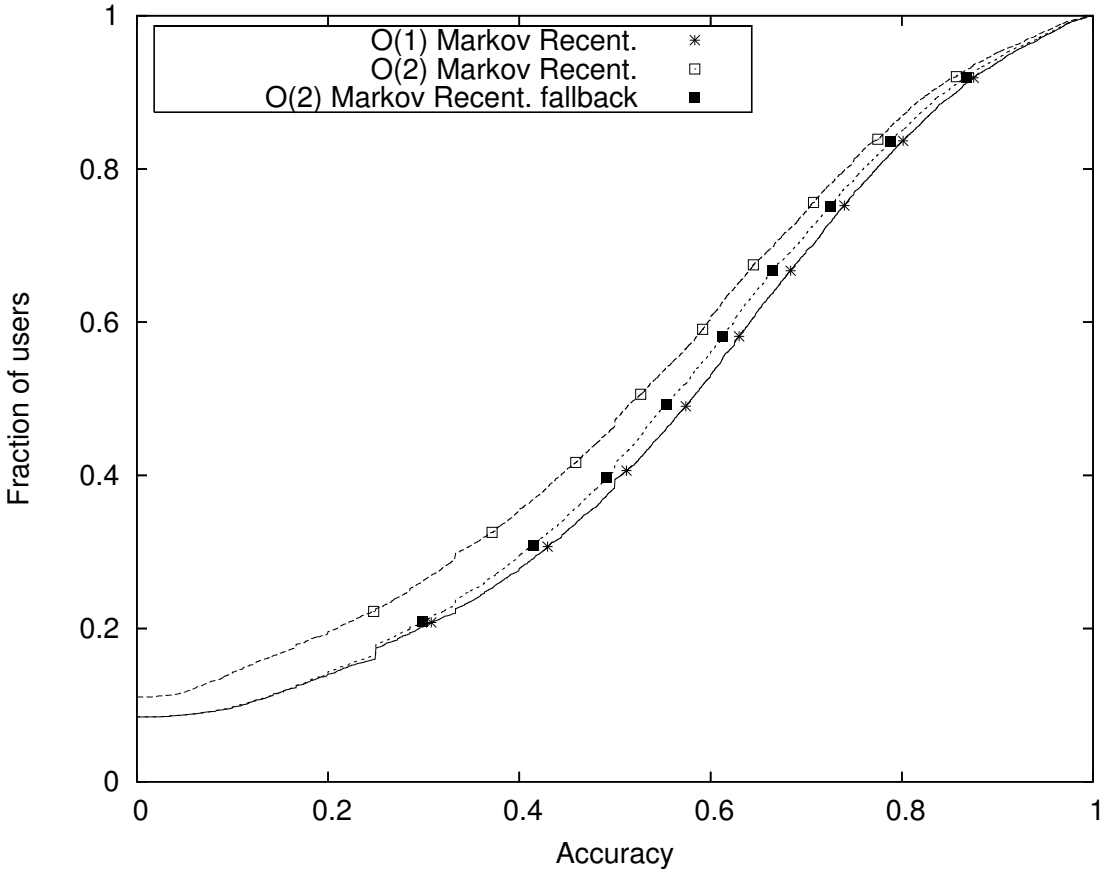


Figure 4.10: Markov using most recent

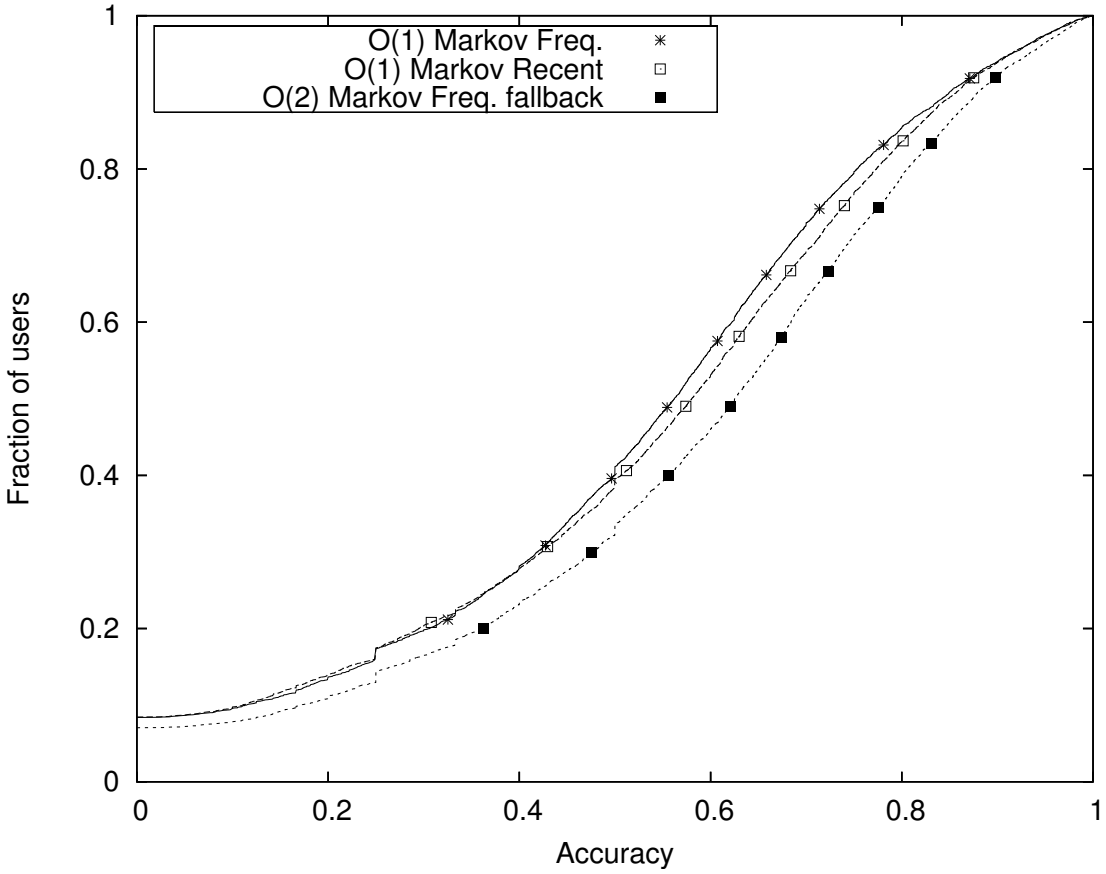


Figure 4.11: Markov: most recent vs. most frequent

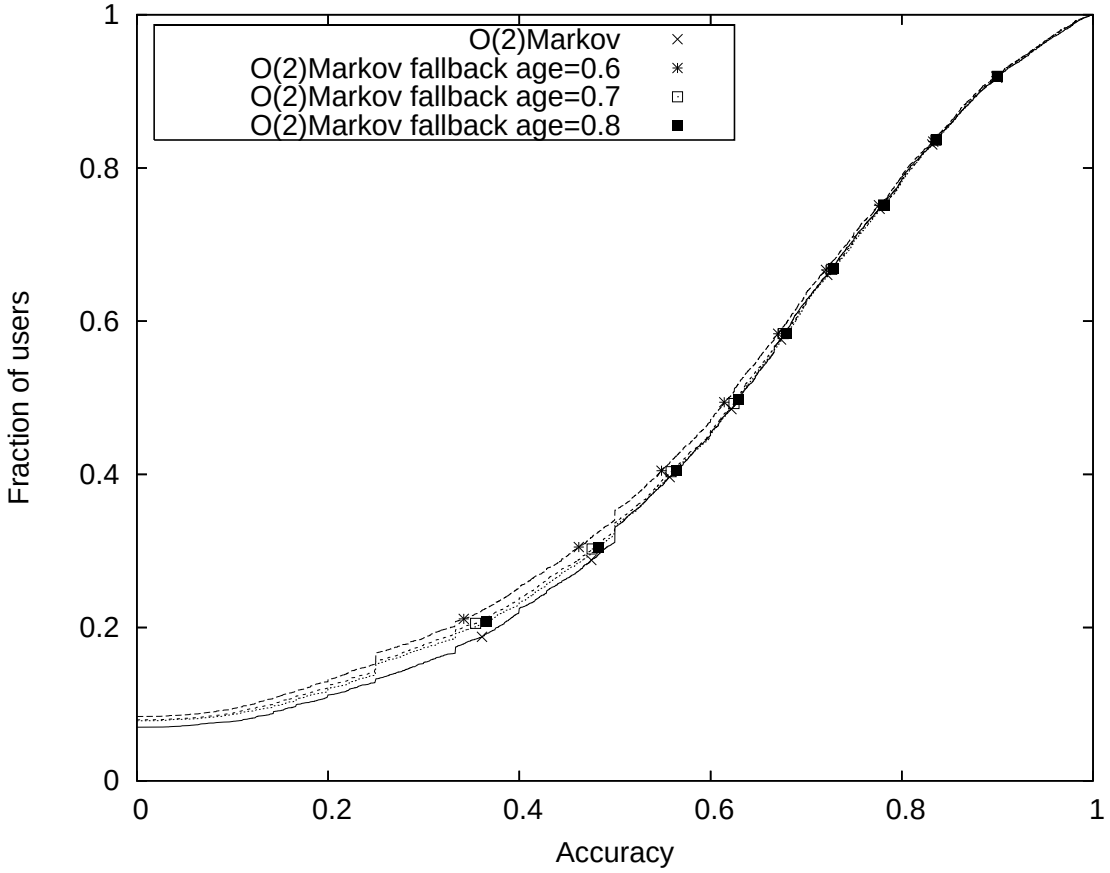


Figure 4.12: O(2) Markov Prediction with exponential decay

4.1.3 LZ-based

We first consider the LZ predictor, shown in Figure 4.13. Since LZ makes no prediction whenever the current context string leads to a leaf in the tree, the plot includes two LZ variants. As an alternative to no prediction (the first curve), we can use the statistics at the root (second curve) to make a prediction based on the probability of each location. That is, when the predictor encounters a leaf, it behaves as if it is at the root and simply predicts the most frequently seen child of the root. Given our accuracy metric, it is always better to make some prediction than no prediction, and indeed in this case the accuracy improved significantly. An even better approach (third curve) is to fallback to progressively shorter substrings of the current context, much as we did with the Markov predictors, until a substring leads to a non-leaf node from which we can make a prediction. This fallback ability is critical to allow prediction to occur while the tree grows, since the trace often leads to a leaf just before adding a new leaf.

Bhattacharya and Das [BD02] proposed two extensions to the LZ predictor. Figure 4.14 displays the performance of the first extension, LZP. Once again, this predictor (as defined in [BD02]) makes no prediction when the context leads to a leaf. We modified LZP to use statistics at the root, which helps, and (better) to try fallback.

The second extension produces a LZ+prefix+PPM predictor nicknamed "LeZi." Figure 4.15 compares the performance of LZ with LZP and LeZi, showing that each extension did improve the accuracy of the LZ predictor substantially.

When we compare the best variant of each LZ form, in Figure 4.16, it becomes clear that the simple addition of our fallback technique to the LZ predictor did just as well as the prefix and PPM extensions combined. (LeZi automatically includes fallback by adding suffix substrings into the tree, so there is no fallback variant.) LZ with fallback is a much simpler predictor than LZP or LeZi, and since the accuracy was similar (and as we show

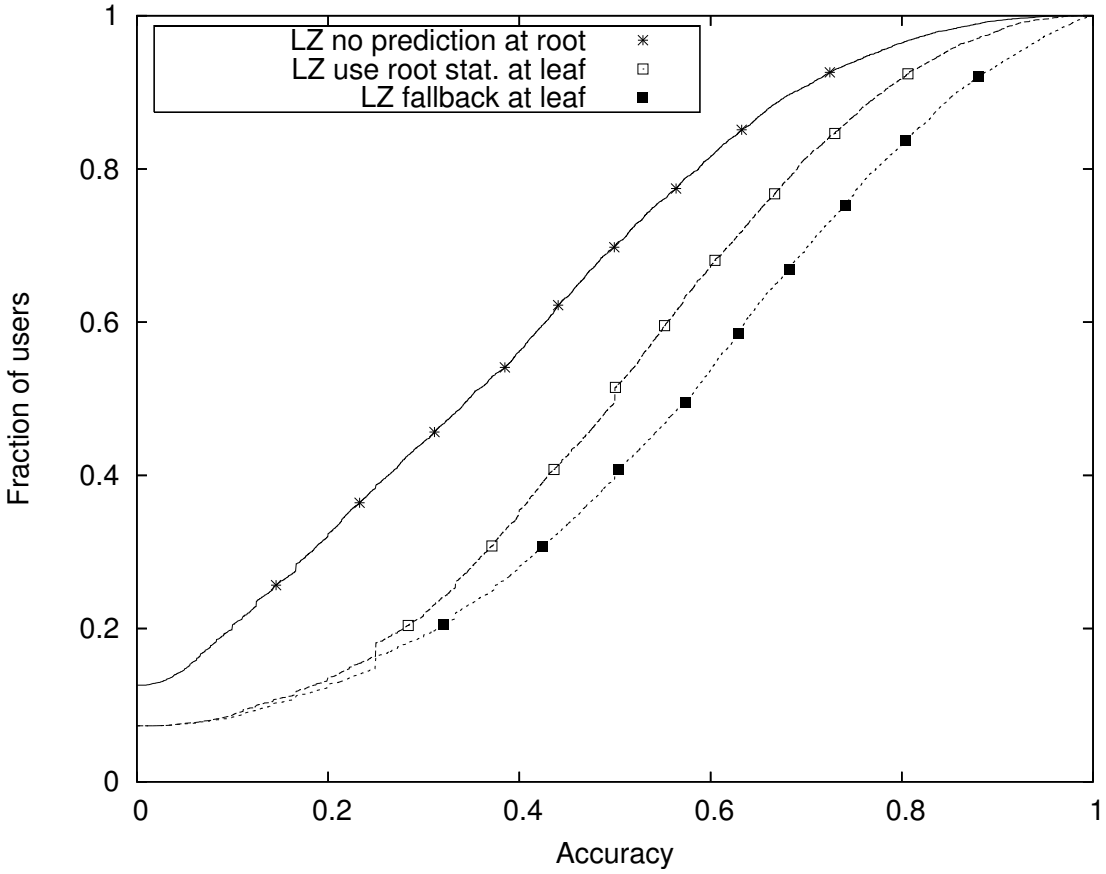


Figure 4.13: LZ predictors

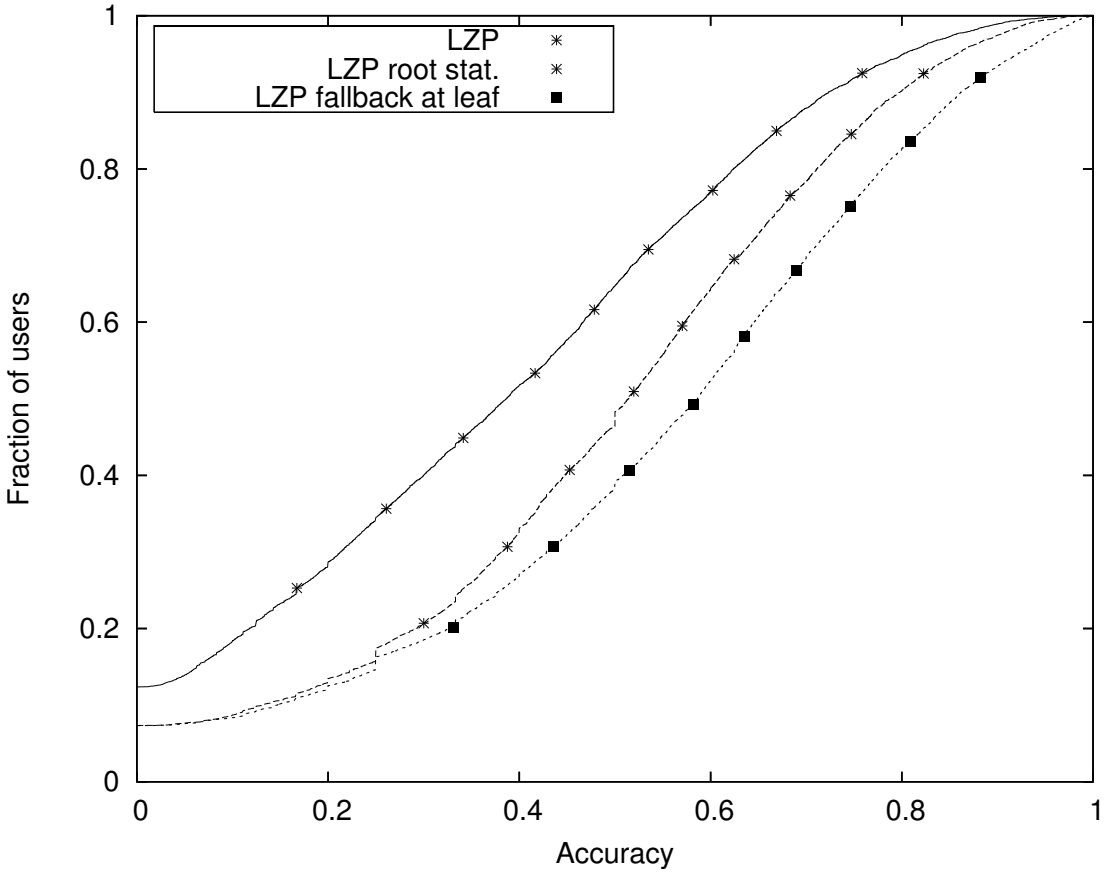


Figure 4.14: LZP predictors

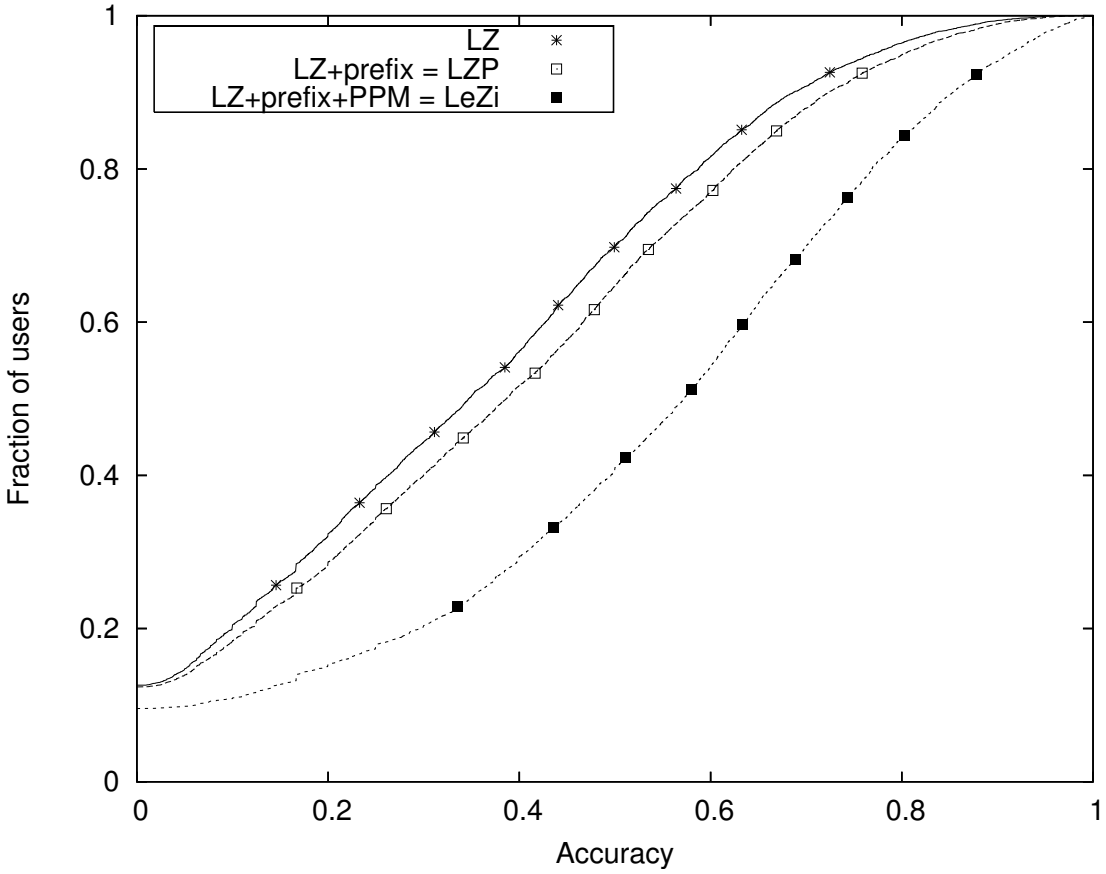


Figure 4.15: LZ, LZP, LeZi predictors

below, the LZ data structure was smaller), among the LZ predictors we recommend LZ with our fallback technique.

4.1.4 PPM and SPM

Prediction by Partial Matching (PPM), like Markov predictors, uses a finite context but blends together several fixed-order context models to predict the next symbol [CT97]. Figure 4.17 shows that the amount of context did not matter much when the order was 2 or above. The higher-order PPM predictors did not have better prediction accuracy.

The SPM predictor [JSA00] is, in a sense, a PPM predictor in which there is no limit on order. Theoretically, the SPM predictor should outperform any finite-order PPM predictor. However, Figure 4.18 shows that both PPM and SPM had an accuracy only slightly better than the $O(2)$ Markov predictor with fallback. For long traces, we found the difference to be negligible. Here we choose $\alpha = 0.5$. We also tried to set $\alpha = 0.25$ and $\alpha = 0.75$; both of the results are worse than $\alpha = 0.5$ as shown in Figure 4.19. The reason may lie here: when we use a smaller α value, we are using a short context with more samples. If we use a larger α value, the number of samples will be smaller. There is a balance between the length of context to be used and the number of samples at that context.

4.1.5 AP prediction overall

We compare the best Markov predictors with the best LZ predictors in Figure 4.20. It is difficult to distinguish the LZ family from the recency-based $O(1)$ Markov, which all seem to have performed equally well. In general, the $O(2)$ Markov predictor with fallback was the best overall. For long traces (Figure 4.21) the median accuracy with this predictor was 72%. It is striking that the extra complexity, and the theoretical aesthetics, of the LZ, PPM, and SPM predictors apparently gave them no advantage.

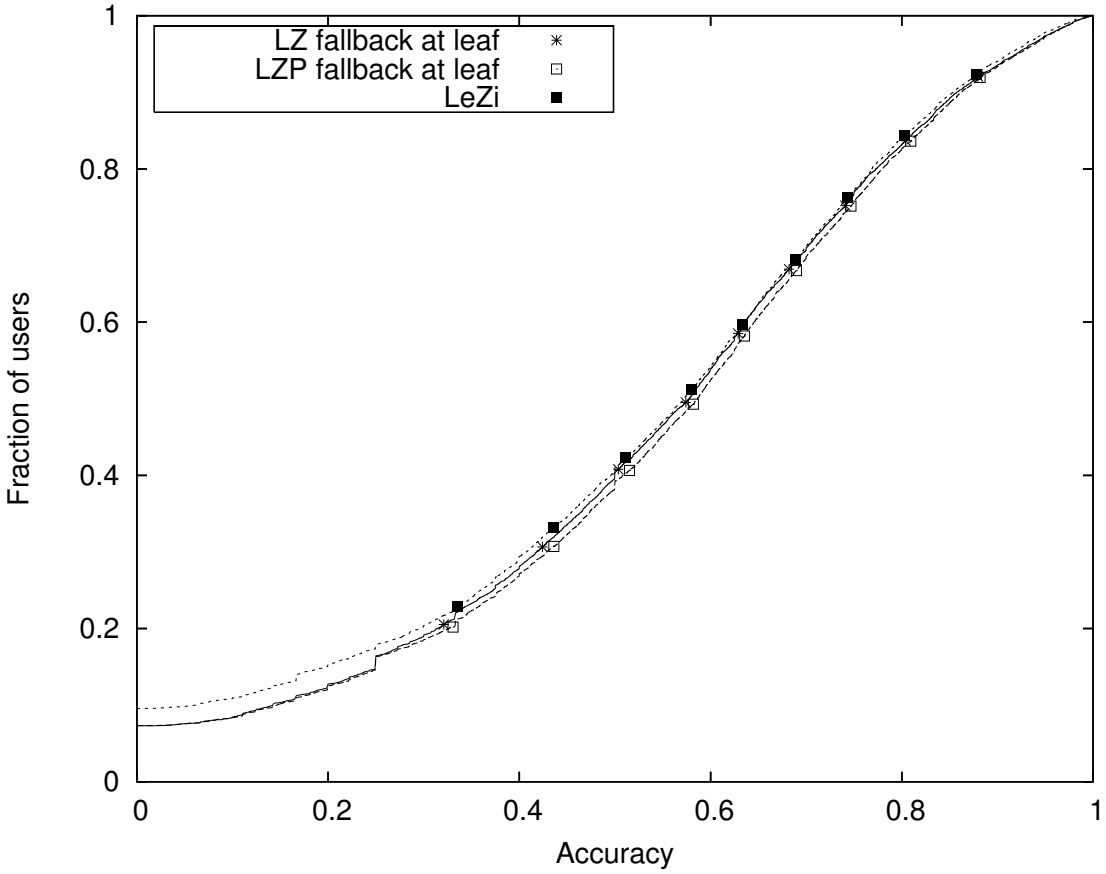


Figure 4.16: LZ predictors with fallback

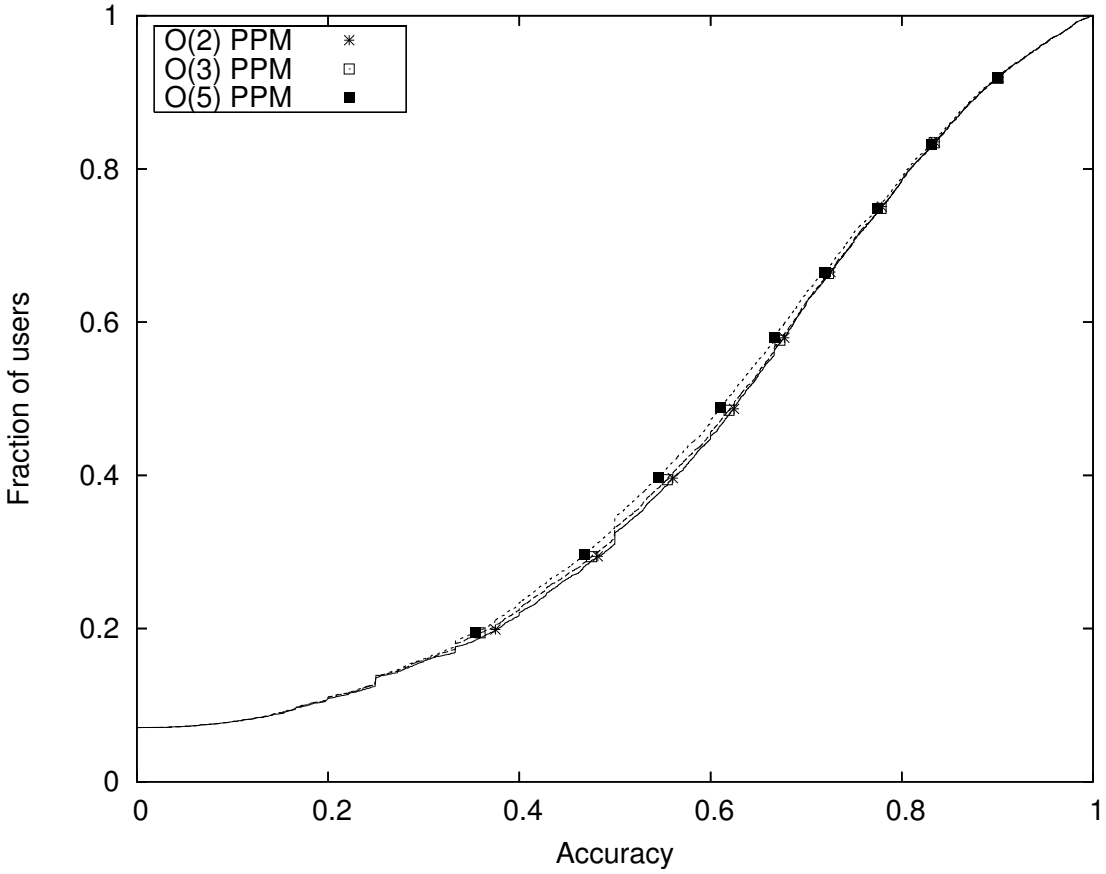


Figure 4.17: PPM predictors

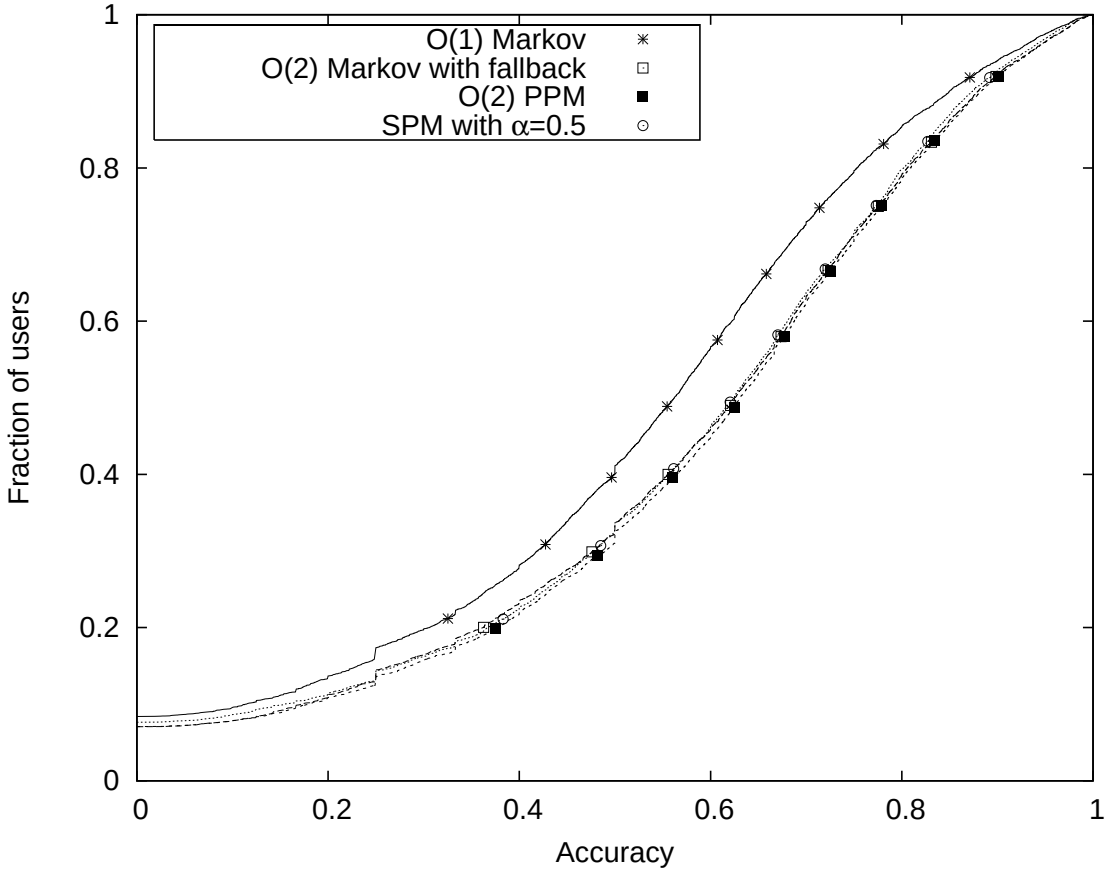


Figure 4.18: SPM predictors with $\alpha = 0.5$

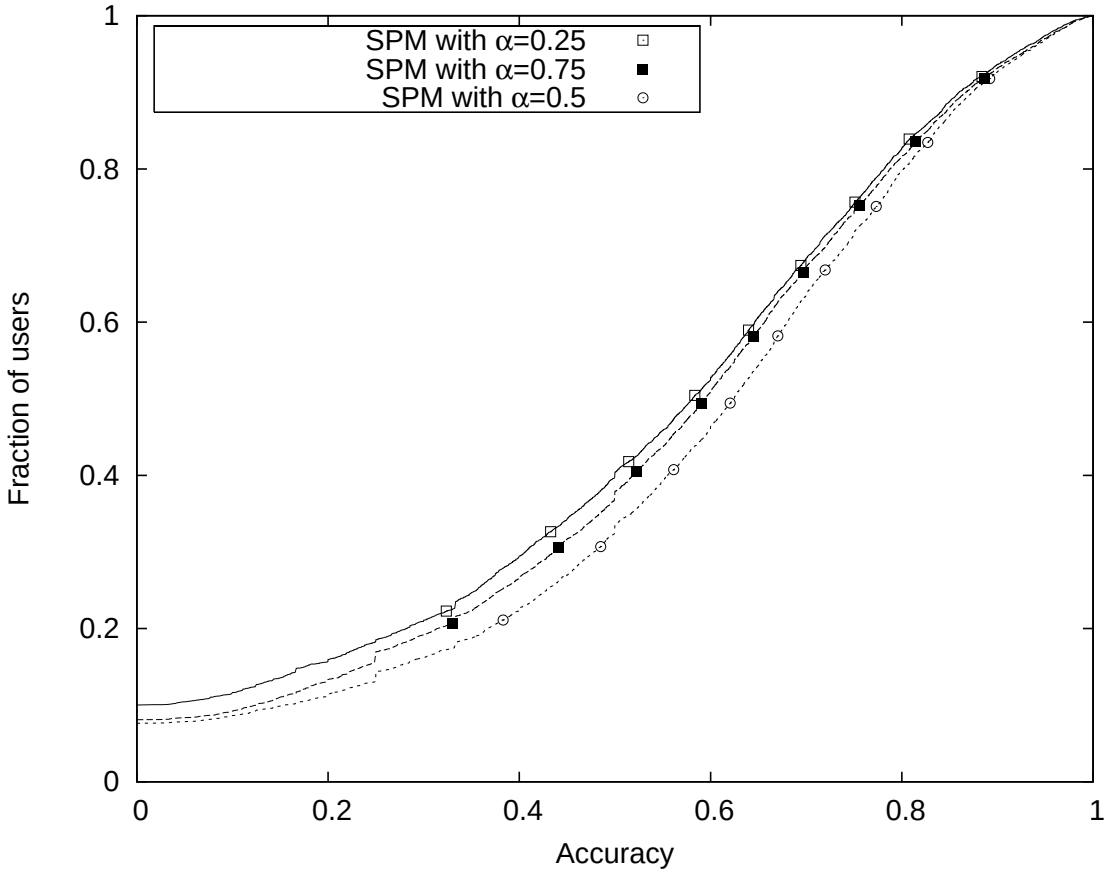


Figure 4.19: SPM predictors with different fractional contexts

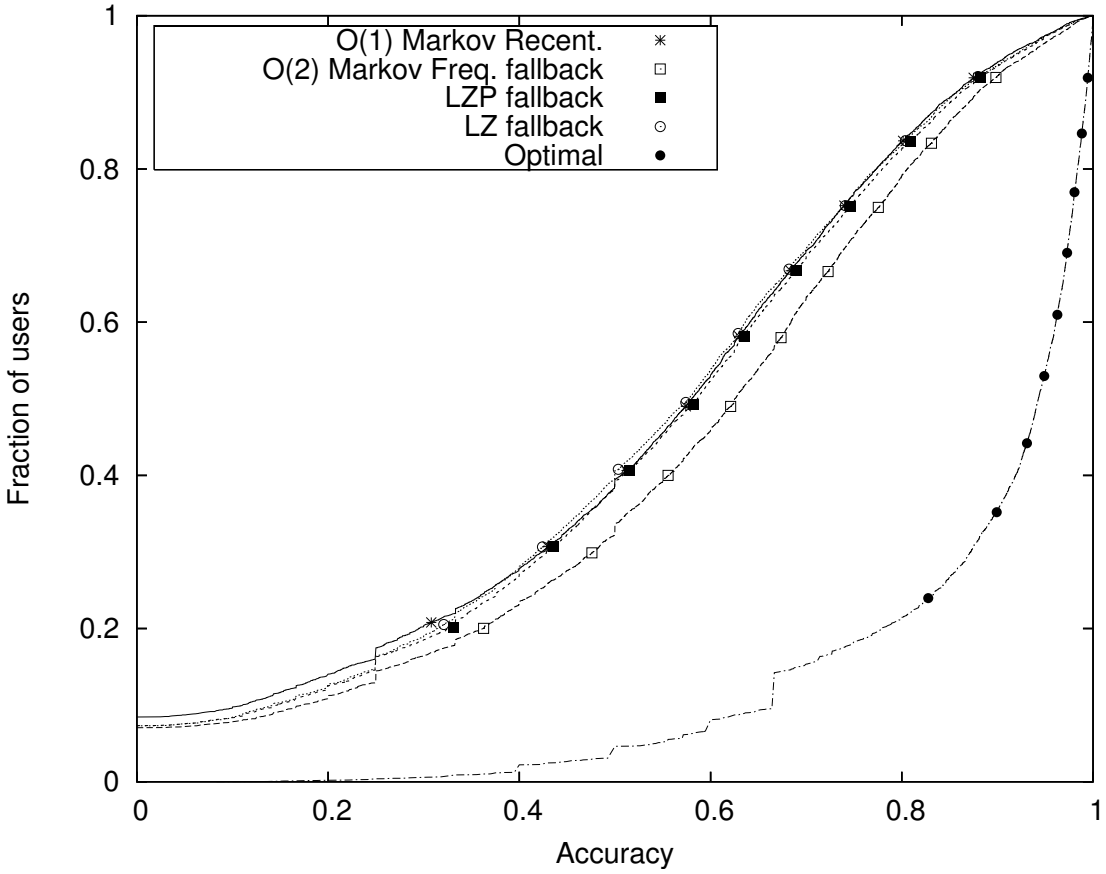


Figure 4.20: The best predictors, compared

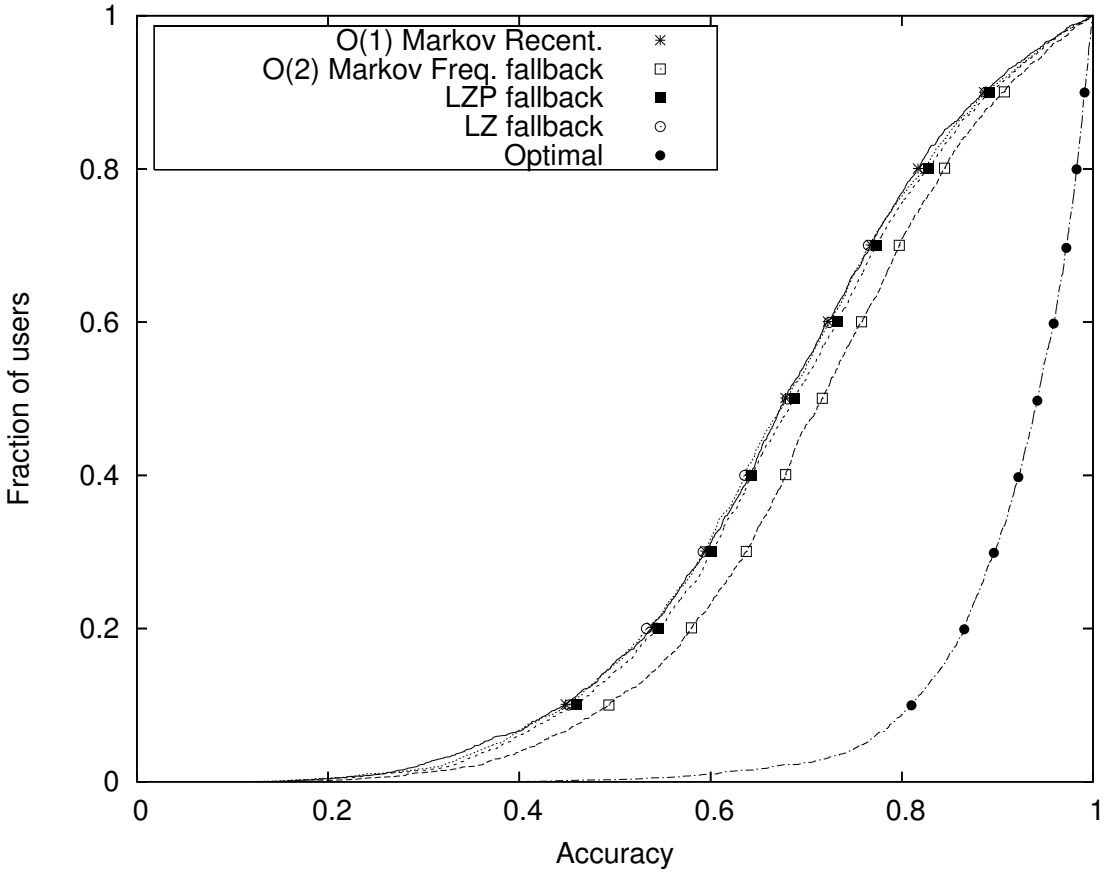


Figure 4.21: The best predictors, compared (long traces only)

We include an "Optimal" curve in Figure 4.20 and 4.21, as a simple upper bound on the performance of history-based location predictors. In our definition, the "optimal" predictor can accurately predict the next location, except when the current location has never been seen before. Although it should be possible to define a tighter, more meaningful upper bound for domain-independent predictors like those we consider here, it seems clear that there is room for better location-prediction algorithms in the future.

4.1.6 Building prediction

In some applications, we may need only a coarser sense of the user's location. We extracted building-level traces from the AP-level traces; a building trace shows the sequence of buildings visited by the user and is necessarily shorter than the original sequence of APs visited. Then we used our predictors to predict the next building in each user's trace. Figure 4.22 shows that the building prediction was more accurate than AP prediction. Figure 4.22 shows only long traces, and shows that the best predictor can approach a median accuracy of 80%. One reason is that there was a smaller number of choices in the building trace than in the AP trace. Note that the sets of users of the two different forms of prediction were not the same; there were 2,190 users who had more than 1000 transitions in the AP-level trace history, while there were only 1,501 users with more than 1000 transitions in their building-level trace. Figure 4.23 shows the results of building prediction for all users. The best predictor had a median accuracy about 70% across all users.

In the building-level prediction, the O(2) Markov with fallback predictor also outperformed the O(1) Markov predictor, by about the same amount as in the AP-level prediction.

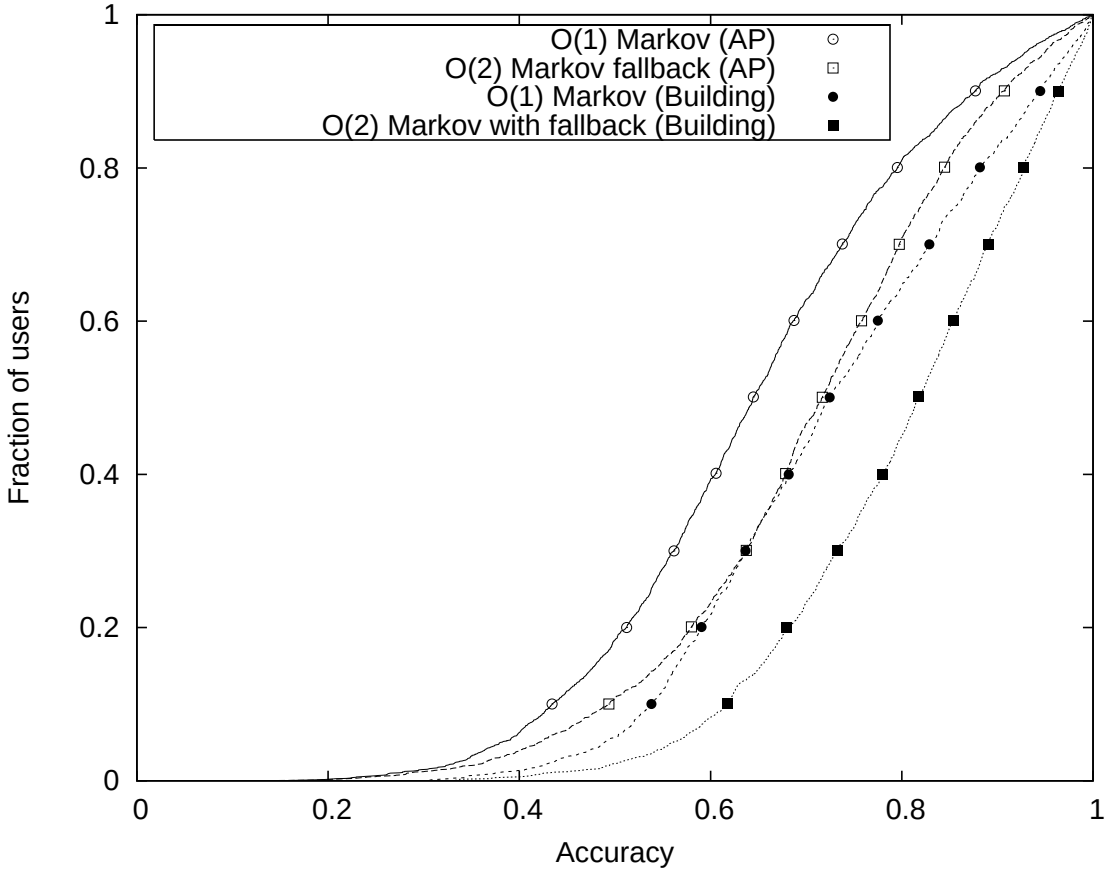


Figure 4.22: Building Prediction (long traces only)

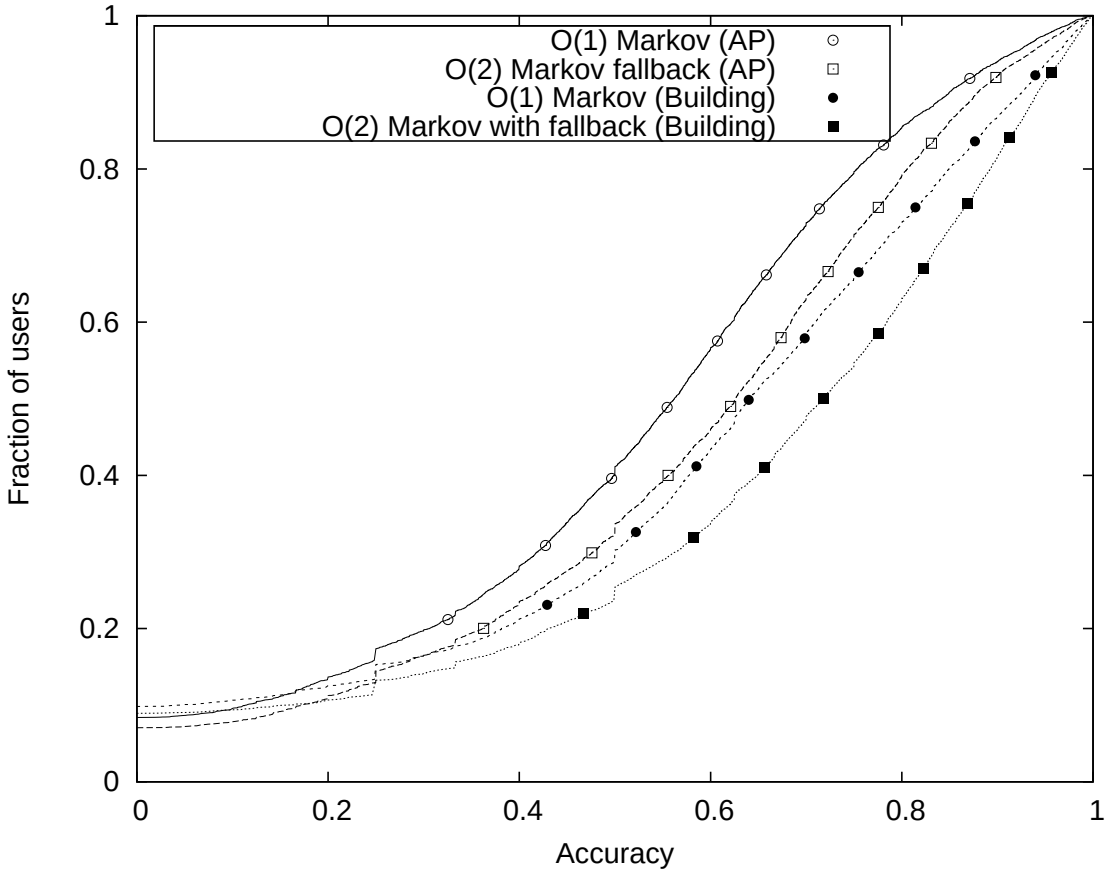


Figure 4.23: Building Prediction (all traces)

4.1.7 Correlation with entropy

It may be that some user traces are simply less predictable than others. Some intrinsic characteristics of a trace may determine its predictability. We explored several such characteristics.

Conditional entropy, as defined in section 3.4.1, seems like a good indicator of predictability. We computed the conditional entropy for each user on their entire trace using Equation 3.10. In Figure 4.24 we compare the entropy of each user, based on the probability table built by the $O(1)$ Markov predictor, with the accuracy of the $O(1)$ predictor for long traces. The correlation is striking, and indeed the correlation coefficient is -0.95 (a coefficient of 1.0 or -1.0 represents perfect correlation). This strong correlation indicates that some users with high entropy are doomed to poor predictability. On the other hand, the correlation is not surprising because the entropy was computed from the same tables used for prediction.

Figure 4.25 shows the correlation of the conditional entropy and the prediction accuracy for all traces. This correlation is not as striking as for long traces. For short traces, the entropy can be low since the amount of information is small and the accuracy can also be low (see below for our comparison of accuracy with trace length).

Figure 4.26 shows the correlation between entropy and the accuracy of the $O(2)$ Markov predictor for all traces; the correlation is visually similar to the $O(1)$ Markov correlation.

4.1.8 Prediction accuracy and trace length

As indicated earlier, trace length has an impact on the accuracy of all the predictors. So far we have regarded accuracy for a given trace as simply the final average accuracy at the end of the trace. At each step i, for each trace that has length at least i, the running average accuracy up to step i is found. There are generally several traces that have length at least i;

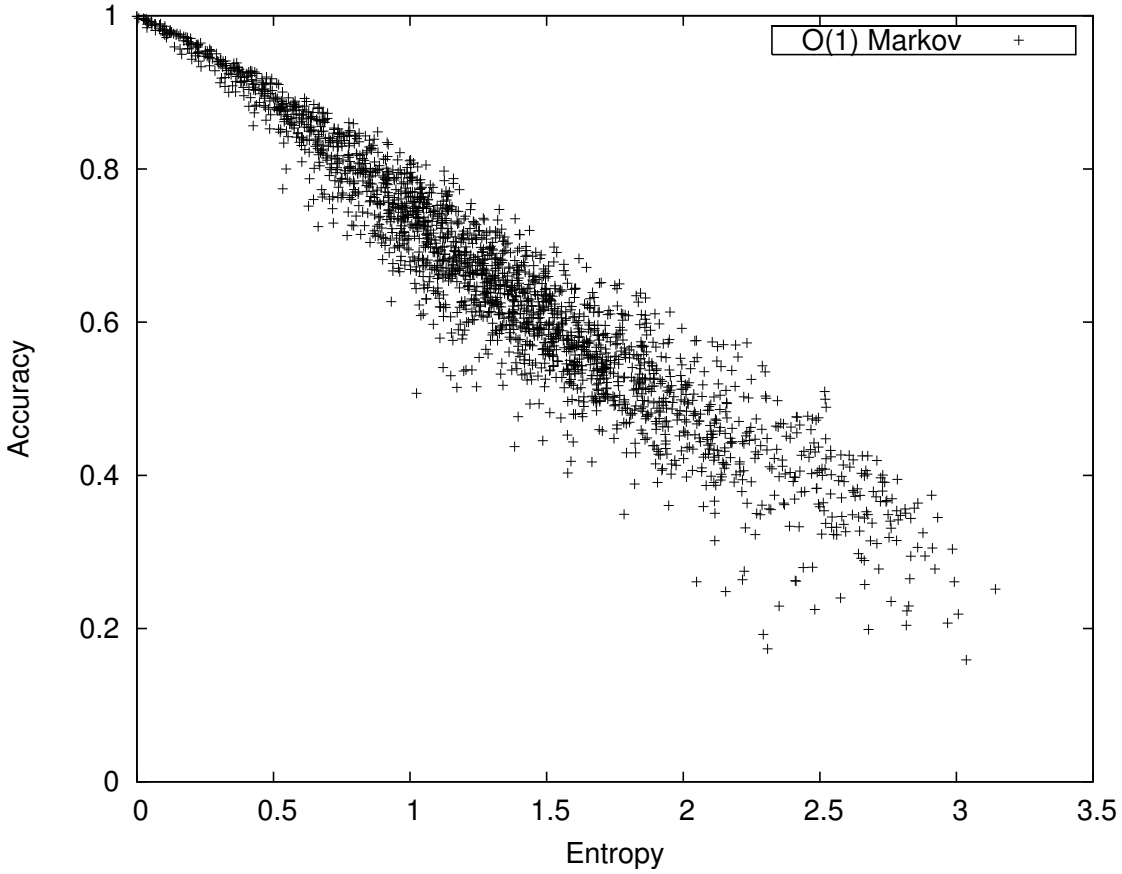


Figure 4.24: Correlating O(1) Markov prediction accuracy with entropy (long traces only)

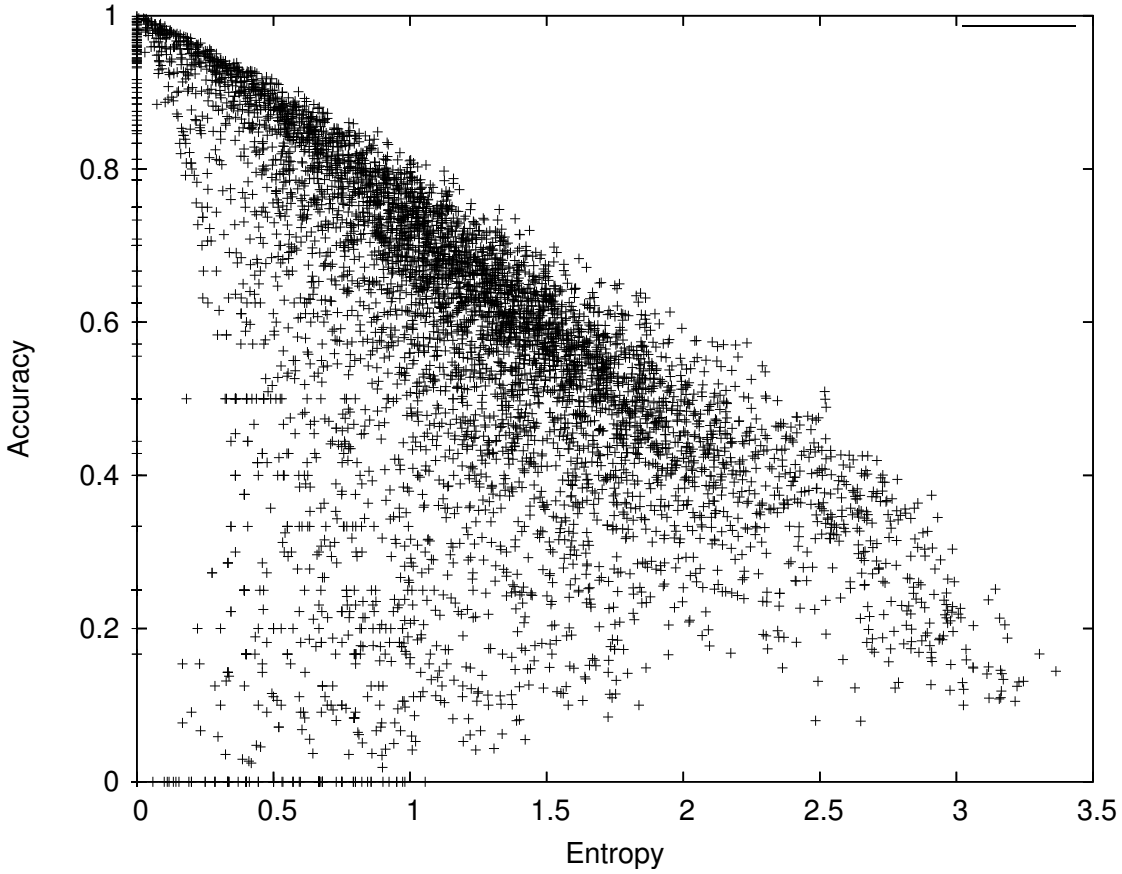


Figure 4.25: Correlating O(1) Markov prediction accuracy with entropy (all traces)

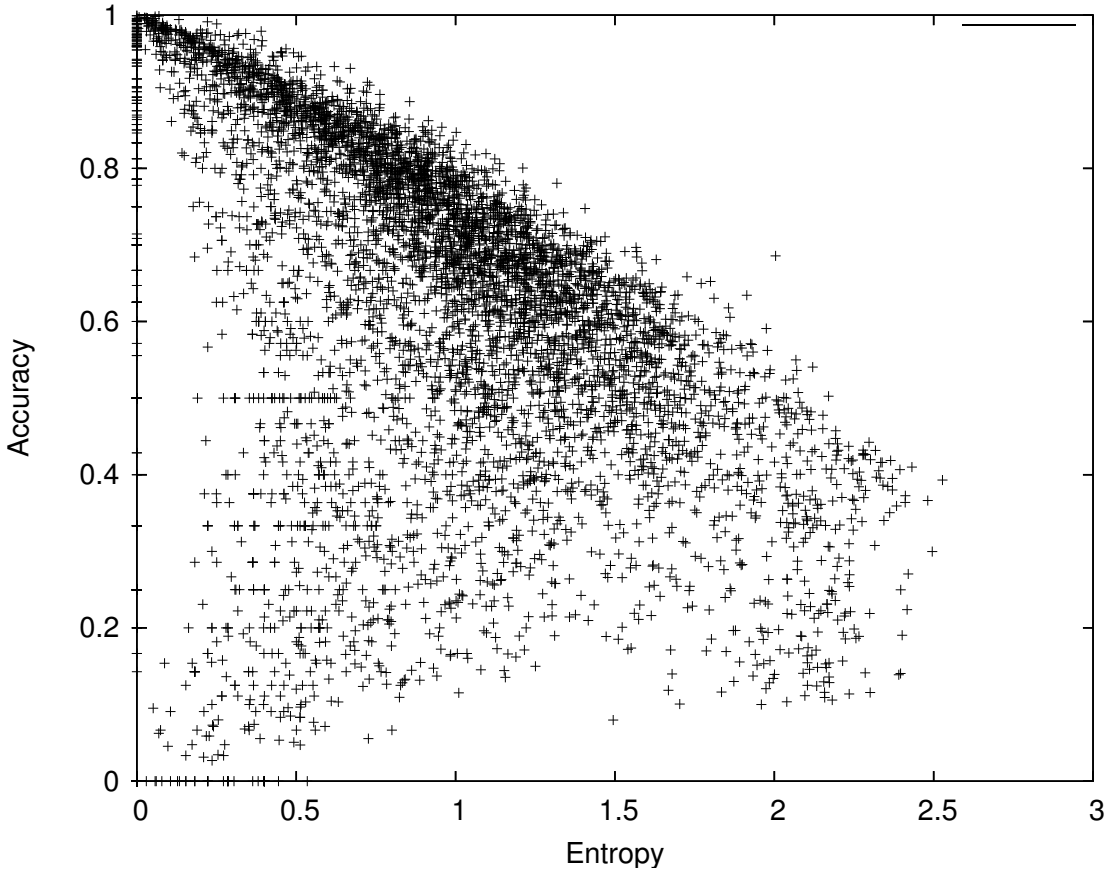


Figure 4.26: Correlating O(2) Markov fallback prediction accuracy with entropy (all traces)

we find the median accuracy at step i among all such traces. This metric, called the *median running accuracy*, enables us to consider how, over all traces, accuracy changes as trace length increases.

Figure 4.27 shows how the median running accuracy increases with trace length for $O(1)$ Markov as well as $O(2)$ Markov with fallback. Clearly the median running accuracy increases rapidly with trace length up to short traces, and then increases only relatively slowly. Note that for this metric $O(2)$ Markov with fallback outpredicts $O(1)$ for all trace lengths.

We tried several equations to fit the experimental data, such as power functions and log functions. We found that the log functions fit best. These curves can be fit to the equations: for $O(1)$ Markov,

$$a(i) = 0.036 \log(i) + 0.4176 \,, \tag{4.1}$$

for $O(2)$ Markov with fallback,

$$a(i) = 0.0367 \log(i) + 0.4647 \,. \tag{4.2}$$

The R^2 of the $O(1)$ Markov fit is 0.8717, and the R^2 for the $O(2)$ Markov fallback is 0.8956. The fits are good for long traces ($i > 100$), but have larger errors for short traces ($i < 100$).

We also investigated the correlation of accuracy and trace length in another way. We had 6,202 users, and each user had a different trace length. Let $u(i)$ be the number of users whose trace had at least i moves. At each step, some users were predicted correctly, while the other users were predicted incorrectly. Let $A(i)$ be the number of users whose prediction at move i was correct. The ratio of correct predictions at move i is $A(i)/u(i)$. Figure 4.28 shows the ratio for $O(1)$ Markov and $O(2)$ Markov with fallback. Actually, this

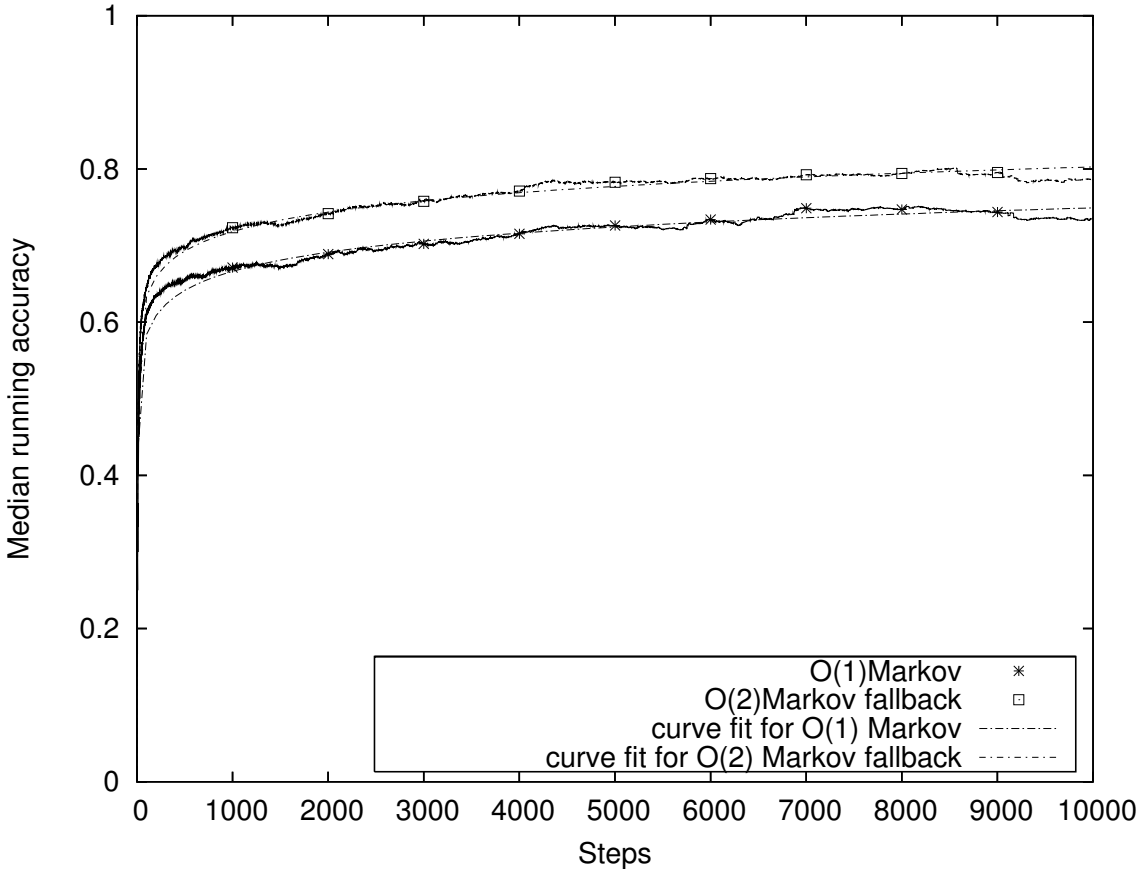


Figure 4.27: Correlating accuracy with trace length

plot smoothes each curve by plotting the median ratio within each bucket of 20 moves. The resulting curves look similar to those in Figure 4.27. Again, the ratio of correct prediction increased rapidly for short traces, then increased slightly for longer traces.

In general, prediction accuracy was quite good for traces with at least several hundred moves, with minimal improvement after that.

4.1.9 Other metrics

Of course, prediction accuracy is not the only metric we could use to compare location predictors. In Figure 4.29 we show the size of the predictors' data structures, at the end of processing each user trace and making predictions.[1] As with the accuracy metric, we plot the table size for each predictor as a distribution across all users. For Markov predictors, we define the size to be the number of entries in the (sparse) transition matrix, except for the recency-based $O(1)$ Markov, which simply needs to record the location of the latest transition for each location ever visited. For LZ predictors, we define the size to be the number of tree nodes. (Since the size of each table entry or tree node is implementation dependent, we do not measure table sizes in bytes. Note that the hidden constant factors of each of the data structures could make a big difference.)

Clearly the recency-based $O(1)$ Markov had the simplest and smallest table, by far. In second place are the $O(2)$ Markov and LZ predictors. PPM used more space, and the LZP and LeZi predictors used by far the most space.

Although the medians of $O(2)$ Markov and LZ predictors appear to be similar, on a closer examination it is clear that the LZ predictor had a much higher maximum. Indeed, this plot is truncated at the right side; for one user, LZ used 17,374 nodes. LeZi used as

[1] The table size of SPM predictor is not shown in the plot. To reduce the software development time, we used an inefficient data structure to represent the user trace. It is not fair to compare this structure with the table size of other predictors. A more efficient data structure can be found in the literature [JSA00].

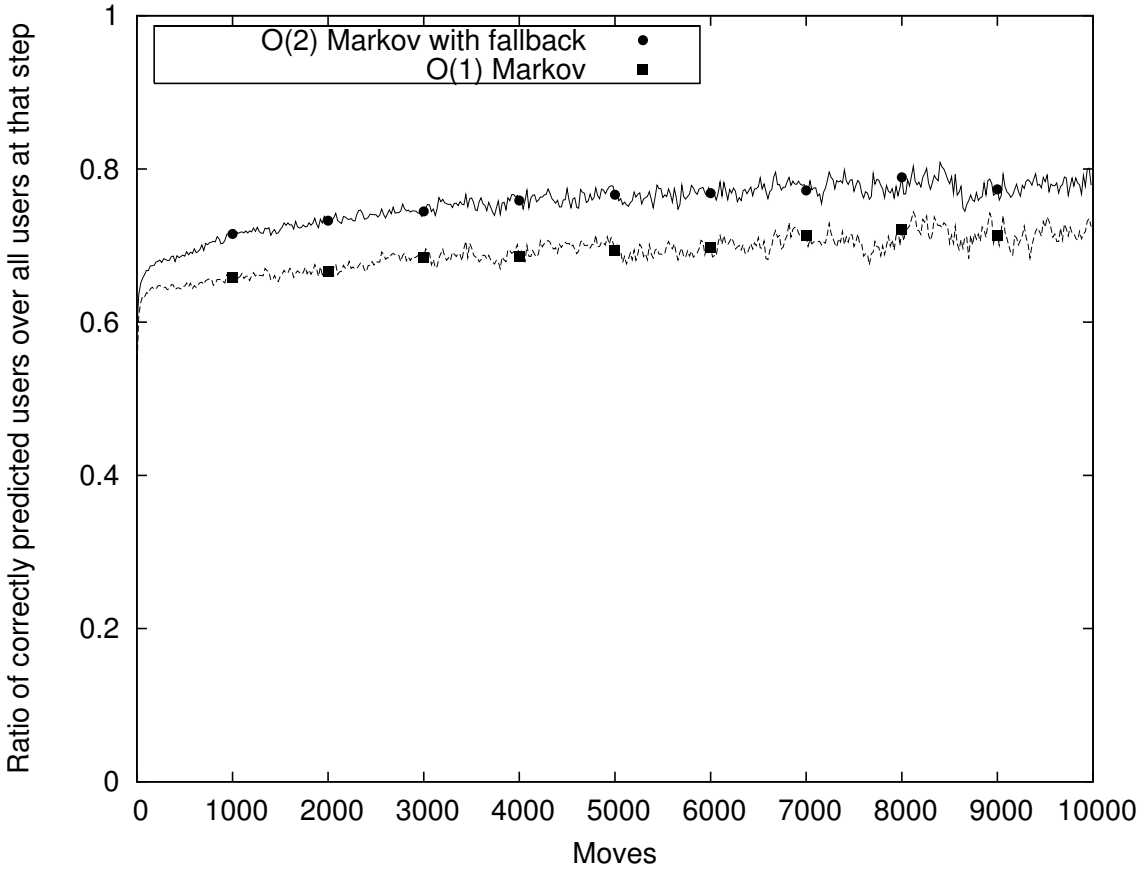


Figure 4.28: Correlating ratio of correct prediction with step

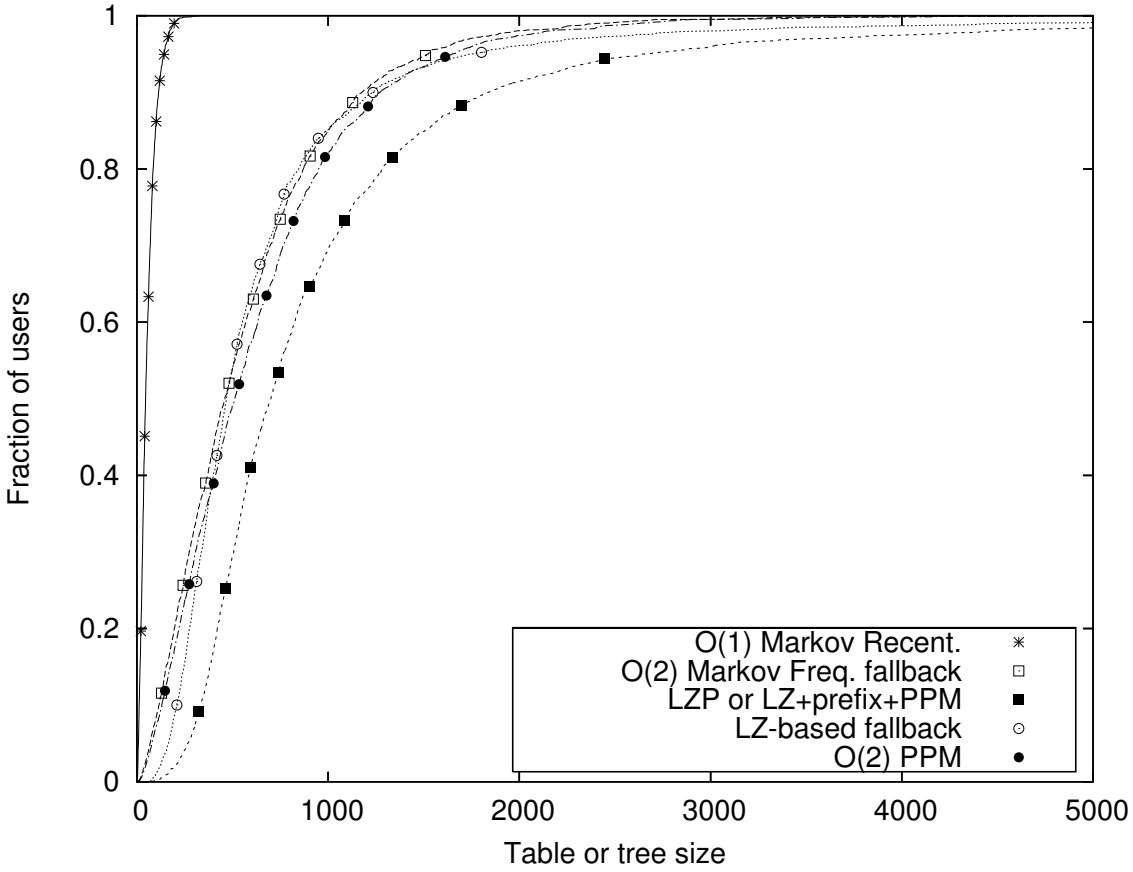


Figure 4.29: Distribution of final table sizes

Table 4.1: Running time per prediction

predictor	time† (millisecond/prediction)
$O(1)$ Markov	0.055
$O(2)$ Markov	0.055
$O(2)$ Markov with fallback	0.056
$O(1)$ Markov Recent	0.054
$O(2)$ Markov Recent	0.054
$O(2)$ Markov Recent with fallback	0.055
LZ	0.055
LZ with fallback	0.056
LeZi	0.058
$O(2)$ PPM	0.086
$O(5)$ PPM	0.099
SPM with α=0.5	3.37
SPM with α=0.25	3.65

† The values are the average of 20 runs.

many as 23,361 nodes! The Markov predictor, on the other hand, never used more than 5,000 table entries.

Computational complexity is another important metric, in some applications; we leave the asymptotic analysis to the theoretical literature. Here, we present our experimental running time for each of the predictors in Table 4.1. The values shown in Table 4.1 were the average of 20 runs, on a Linux computer with 2 2.80GHz Pentium-4 CPUs and 2 GB memory. Note that we evaluated the running time of these predictors on our data (e.g., average milliseconds per prediction), but we have not yet tuned our implementations (we used a slow string-matching algorithm for SPM). The simpler Markov predictors had running time similar to the LZ predictors, and both were faster than PPM and SPM predictors.

4.2 Joint Time and Location Prediction

In this section, we examine the time predictors and the joint-time-and-location predictors. In evaluating these predictors, we used a two-month trace that was collected from October 2003 to November 2003.

Note that, in these experiments, we did not count the metrics for moves from and to OFF. There were many stationary users in the network. They just got on and off the network, and did not change APs when they were on. We trained our predictor with the October trace data and counted the metrics only for November. A few users appeared in October, but never showed up in November, so it is possible that training our predictors on the history of handoffs of these users may contribute negatively to the aggregated profile.

4.2.1 Time of day prediction

We first used the $O(k)$ Markov predictor to predict the time of day, e.g., the hour of the day, for the next handoff; each quantized time value represents a symbol in the history. The interval size (e.g., one hour) determines the number of states, hence the complexity of the Markov model. Figure 4.30 shows the handoff-time prediction results using this Markov time of day predictor. We also implemented a fallback mechanism that uses lower-order Markov predictions whenever a higher-order predictor did not have enough history for the computation. The results show that the fallback helped the predictor.

There are two sets of curves in Figure 4.30. The upper set of curves are the location-dependent predictions, which means the predictor used only the history from the user's current location. These curves show low accuracy for both $O(1)$ and $O(2)$ Markov predictors, even with the fallback mechanism. The median accuracy of the $O(2)$ Markov predictor with fallback was about 15% higher than the $O(2)$ Markov predictor without fallback. Figure 4.30 shows that all the location-dependent Markov predictors predicted the

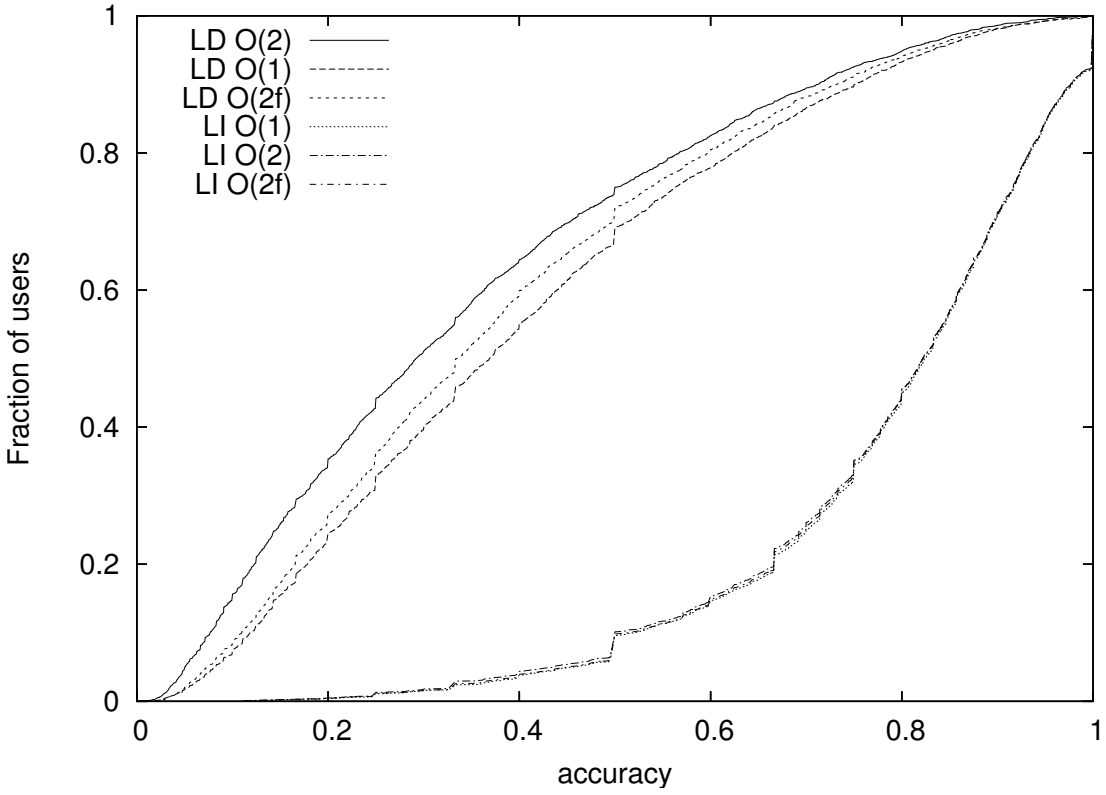


Figure 4.30: Accuracy of time of day prediction using $O(k)$ Markov predictors. The time of the day is quantized into one-hour intervals. The key "LD" means location dependent and "LI" means location independent. The notation $O(2f)$ means $O(2)$ Markov with fallback.

user's moving time with median accuracies less than 40%. The higher-order Markov predictor without fallback did not perform as well as lower-order Markov predictors. The $O(1)$ Markov predictor performed almost as same as the $O(2)$ Markov predictor with fallback.

The lower set of curves are location independent. These predictors used the history without regard to location. The location-independent Markov predictors had higher prediction accuracies than the location-dependent ones. There was not much difference between $O(1)$ and $O(2)$ Markov predictors. The median accuracies of both predictors were near 80% for the one-hour interval.

When we used a smaller interval, say 10 minutes, the accuracies were lower. The median accuracies for location-independent prediction, as shown in Figure 4.31, were about 60%. The $O(1)$ Markov predictor beat the $O(2)$ Markov predictors, even with fallback. This indicates that using longer history of handoff times does not help handoff-time prediction.

Large intervals improve accuracy but degrade the precision to a point that may not be useful for some applications. Although it may be useful to predict the time of day for the next handoff, the one-hour interval may seem too large. Even a precision of 10 minutes is not acceptable for some applications. In summary, the Markov time of day predictor is not able to predict the time precisely.

4.2.2 Duration prediction

The user's movement time is hard to predict. It may be easier to predict how long a user will stay in his current location. Intuition suggests that people tend to stay at APs in specific patterns. One specific behavior we expected was a similarity in the residence times. We first experimented with Markov predictors to search for stationary patterns in the user history. Note that, to conduct Markov predictions, we needed to quantize the durations as symbols.

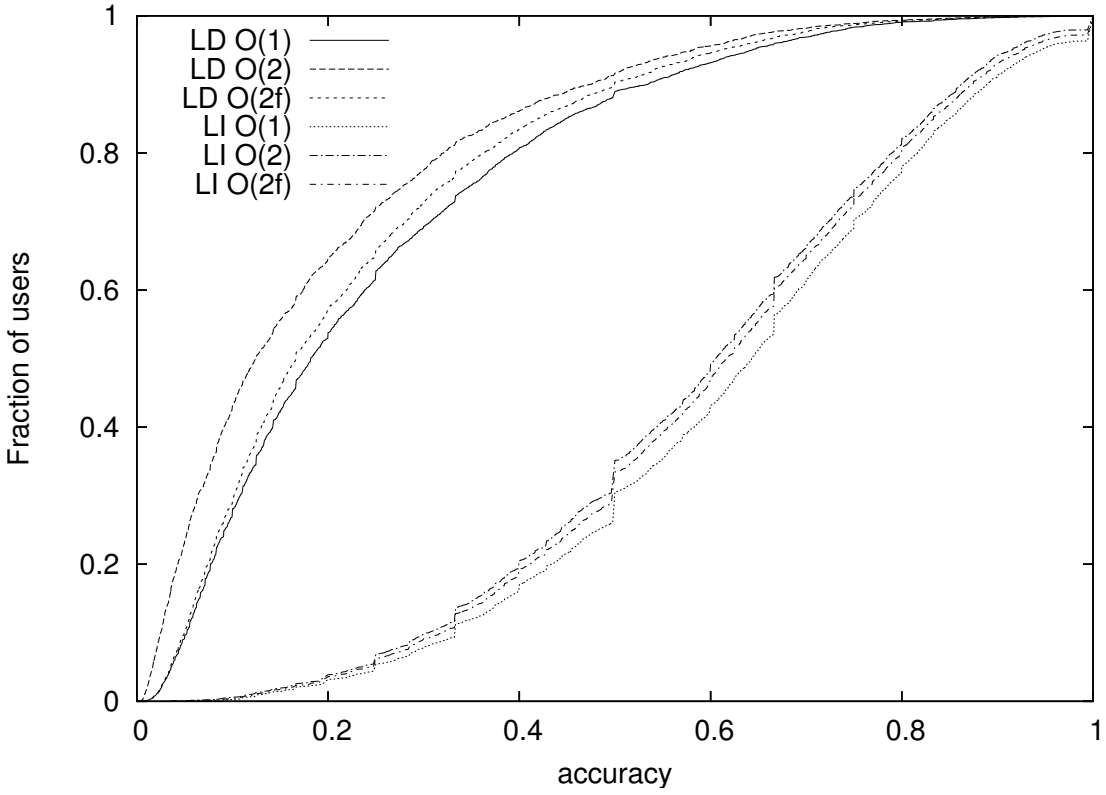


Figure 4.31: Accuracy of time of day prediction using $O(k)$ Markov predictors. The time of the day is quantized into 10-minute interval. The key "LD" means location dependent and "LI" means location independent. The notation $O(2f)$ means $O(2)$ Markov with fall-back.

Figure 4.32 explores the performance sensitivity against different orders of dependency on the previous durations as well as the sensitivity to the quantization interval. In this figure the accuracy metric is the ratio of the number of correct predictions to the number of all attempted predictions. Similar to location prediction, a prediction is correct if the predicted value was in the same interval as the actual duration; otherwise, it is incorrect. The results demonstrate that a 10-minute interval was about 20% less accurate than a one-hour interval for the same order Markov predictor. The rightmost curve, i.e., the O(2) Markov predictor with fallback using an one-hour interval, is the best with a median value of 0.88. The worst is the O(2) Markov predictor without fallback using 10-minute interval, with a median of 0.53. It is not surprising that the bigger interval led to higher accuracy, because there was a higher probability that the actual duration falls within the interval. Moreover, with a bigger interval the predictor had fewer states, reducing the chance of choosing the wrong state. The lagging performance of O(2) with no fallback is because we count the cases where predictor cannot find a similar previous context as unsuccessful predictions. Yet, the fallback results suggest that whenever there is sufficient context, O(2) provides better accuracy than O(1). The location-dependent Markov duration predictors were less accurate than the location-independent Markov predictors. Overall, the Markov predictor fared poorly at predicting duration.

Now consider the moving-average predictor. Since this predictor can operate on continuous variables, we do not need to quantize the input to the predictor. However, the accuracy metric itself requires that we quantize the output of the predictor. With a quantization interval of one hour, we found that the median accuracy varied from 85% to 90% for the experimented prediction orders and dependencies, which is higher than the location independent Markov time of day predictor with the same one-hour interval. Figure 4.33 explores different orders of moving-average predictors as well as the location dependence. Although the curves are close to each other, the accuracies of the location-dependent moving-average

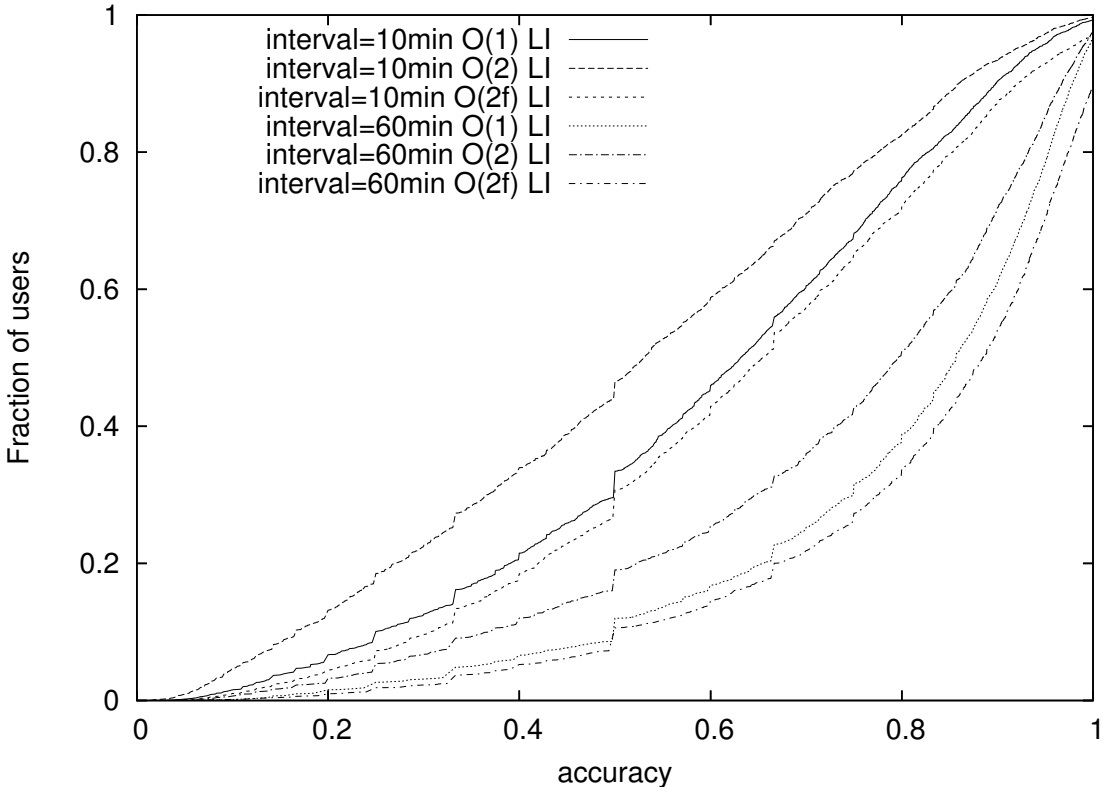


Figure 4.32: Duration prediction with based Markov predictor. The duration is quantized using one-hour intervals and 10-minute intervals. The notation $O(2f)$ means $O(2)$ Markov with fallback.

predictors were slightly worse than the results of the location-independent predictors. For both cases, using a one-hour interval, the curves of the results cross over and are close. However, when using a 10-minute interval (Figure 4.34), the higher-order predictors performed better than lower-order ones; whereas, the location-independent predictors still are better than location-dependent ones. Better performance of the location-independent model suggests that a user spent similar amounts of time at different APs in her subsequent moves, while the residence time at the same AP exhibited larger deviations.

Although accuracy is a sensible measure for discrete input/output predictors such as the Markov predictor, it is not as suitable for continuous input/output predictors. Many applications want to know how predictions deviated from the actual value. Therefore, we used the earliness and lateness metrics to measure the error of the duration prediction.

Figure 4.35 compares three predictors in terms of the *earliness* using each of their best observed results within a set of parameters that we explored. Clearly, the Markov duration predictor was superior. The CDF duration predictor was slightly less accurate than the moving average duration predictor. Nonetheless, all of these predictors failed to predict duration accurately for many users. The median user experienced an average earliness of 604, 1156, and 1218 seconds, for location-dependent $O(2)$ Markov duration predictor using 10 minutes interval, $O(1)$ location-independent moving average predictor, and CDF duration predictor using probability 0.1, respectively. The 80% percentile user had an average earliness of 1662, 1690, and 4479 seconds, for the above three predictors, respectively. Although these average earliness values may be skewed by exceptionally early predictions, many users had higher averages and the range was quite broad. Still, for some applications it may be possible to use these predictions.

We also explored the lateness metrics for all the above four predictors. We found similar broad distribution and extremely high mean lateness. The median user had mean lateness of 1000 to 10,000 seconds. Figure 4.36 shows the best observed curves of each of their

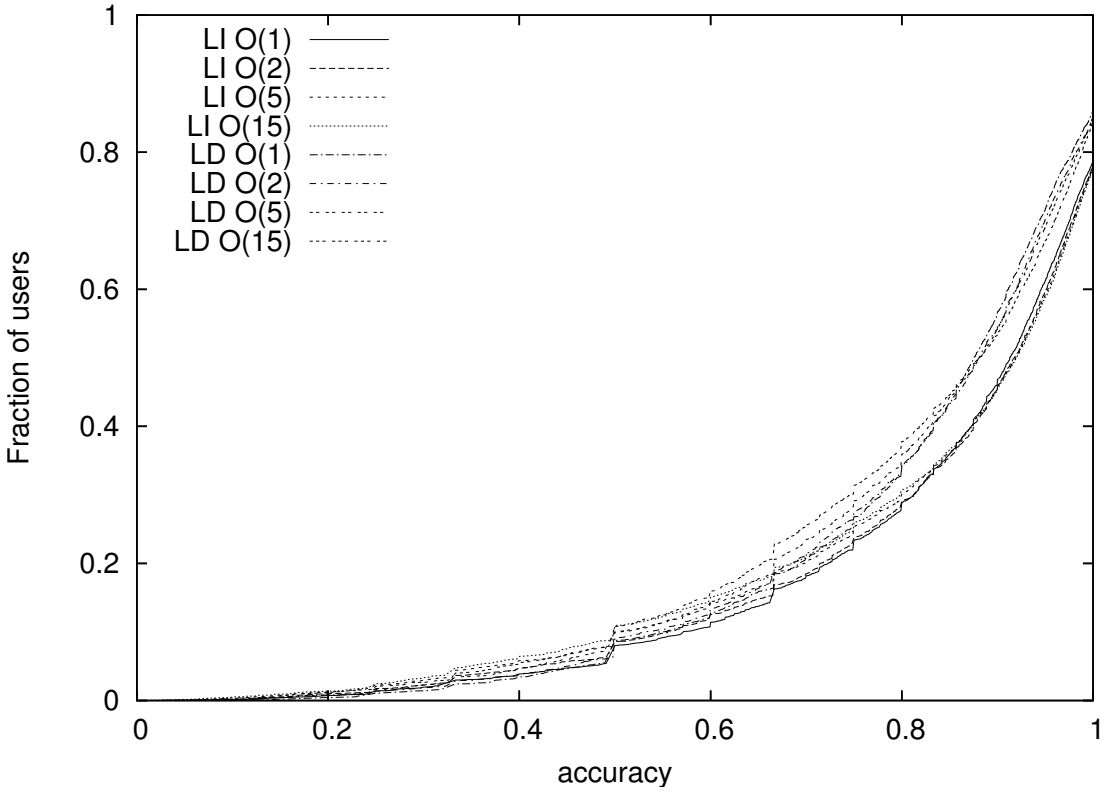


Figure 4.33: Duration prediction using moving average predictors. The prediction output is quantized into one-hour intervals to compare with the actual duration.

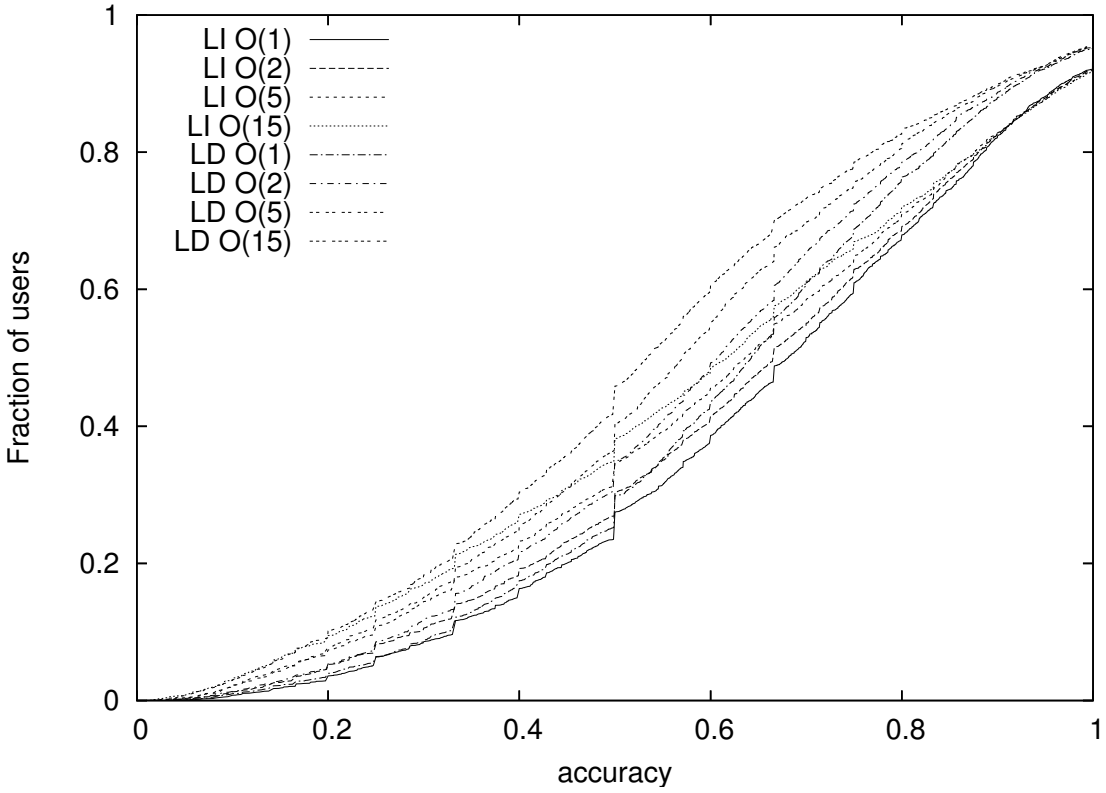


Figure 4.34: Duration prediction using moving average predictors. The prediction output is quantized into 10-minutes intervals to compare with the actual duration.

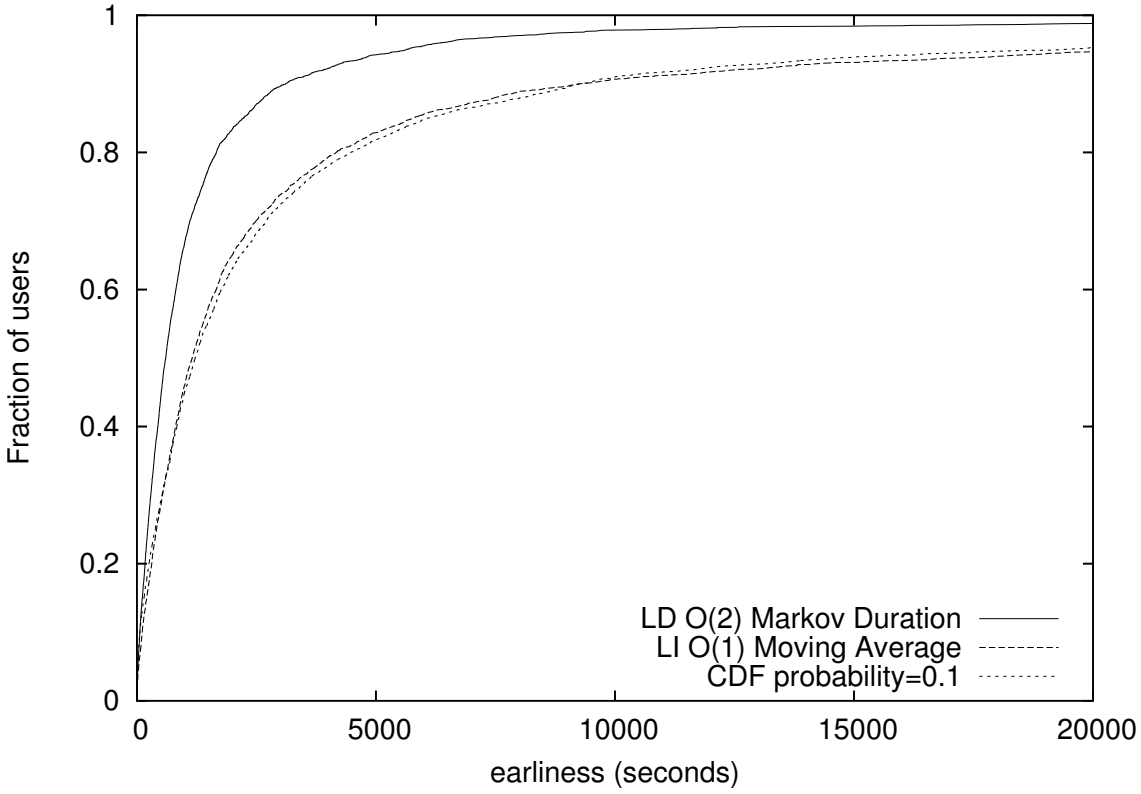


Figure 4.35: Comparison of earliness measurement of predictions: Markov duration prediction, CDF duration prediction, and moving average duration prediction. The curves in the plot are the best observed curves of each of their predictors, across a set of parameters we explored. We truncated at 20,000 seconds. Here, curves toward the left are better. That differs from the previous cumulative plot.

predictors, across the set of parameters we explored. The lateness numbers were even larger than the earliness numbers.

The accuracy, earliness, and lateness metrics indicated that a precise time prediction is not a realistic goal over our real WLAN traces. This fact motivated us to investigate predictors with "soft" outputs that represent the likelihood of the events rather than their exact values, e.g., what is the likelihood of leaving the current location within 5 minutes? Such likelihoods in time prediction may depend on how long the user has stayed in his current location as well as the past history and predicted future location. Since the accuracy, earliness, and lateness metrics are not suitable to measure the quality of such predictions, we use the under-provision and over-provision metrics to quantify the errors in probabilities. In what follows, we present our results for joint time and location prediction, i.e., MarkovCDF predictor, using under/over-provision metrics.

4.2.3 Joint Time and Location Prediction

For joint time and location prediction, we combine the Markov location and the CDF duration predictors. This joint predictor returns a list of possible next locations with the probability for each location, to which the user may move within a specified time.

In Figures 4.37–4.40, we show the under-provisioning and over-provisioning per user and per AP, where the latter helps us to understand whether the predictability is an attribute of the location or of the user. Results reflect the performance differences for different prediction orders and for the cases when individual or aggregate histories are used. The individual predictor used only this user's prior handoff history, while the aggregate predictor used the combination of all users' prior handoff histories. In these experiments, the predictions are updated at the end of every 60-second time slot, computing the handoff probability for the next 60-second interval. As a comparison, we also show the results of

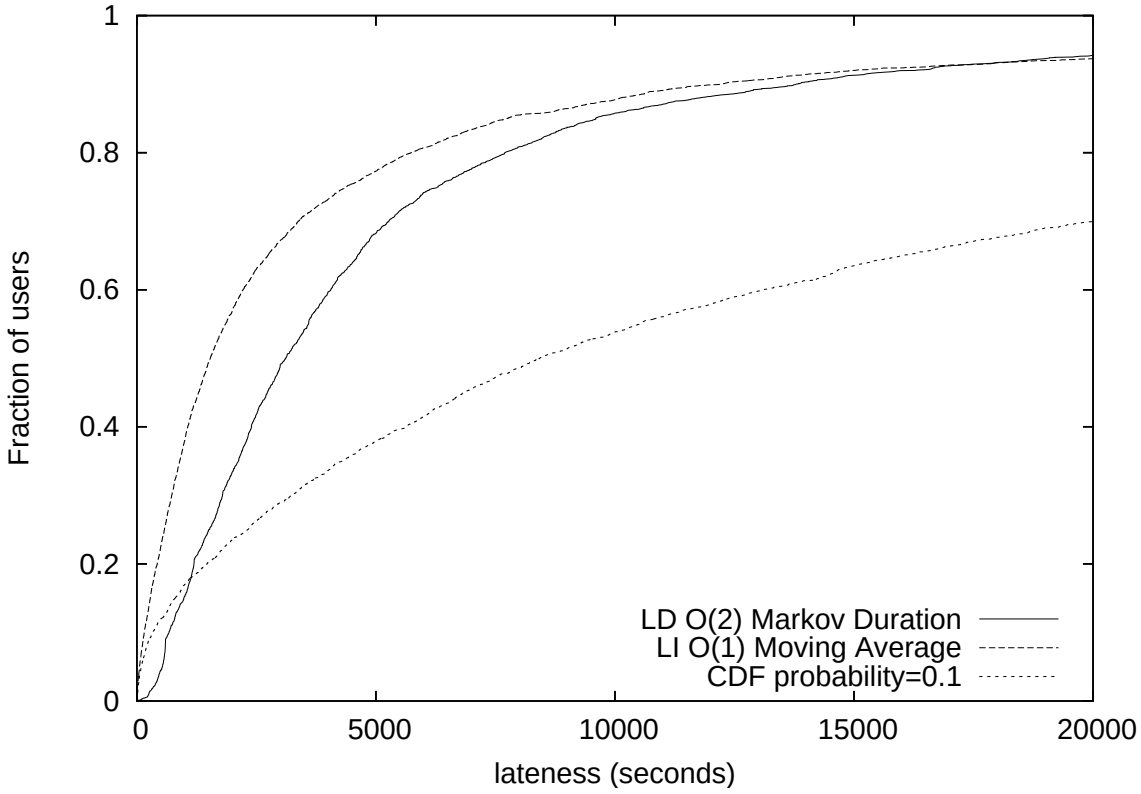


Figure 4.36: Comparison of lateness measurement of predictions: Markov duration prediction, CDF duration prediction, and moving average duration prediction. The curves in the plot are the best observed curves of each of their predictors, across a set of parameters we explored. We truncated at 20,000 seconds. Here, curves toward the left are better.

the NG predictor in these plots. We use one month's traces to construct a directed graph representing transition history and then use the graph for the NG predictor to predict the probabilities on the second month's traces.

For both under-provision and over-provision, lower values (upper curves) represent better performance. Figure 4.37 shows the CDF plots of average under-provision for users and Figure 4.38 shows the CDF plots of average under-provision for APs. The aggregate-profile predictors used all users' history information so that if a particular user did not visit an AP before, the predictor could still make a prediction based on the movements of other users at that AP. The probabilities spread out over many locations, perhaps far more than a single user usually visited. Furthermore, the probabilities depended on how long the user had stayed at the current location. Thus, on one hand, for a given time, the predictor might predict the user would move with a low probability. On the other hand, the probabilities might also be skewed by some extreme users who bounce back and forth among several APs. Therefore, the provision by the aggregated predictor at the location where the user actually moved was too little. In contrast, the individual-profile predictor returned probabilities that reflected the user's personal handoff pattern, so the user was usually better provisioned.

The NG predictor can predict a location that the user had not visited before, but it did not consider how long the user had stayed in the location. It always expected the user to move within the given period. As a result the NG predictor made higher under-provision than the individual-profile MarkovCDF predictors. Although the NG predictor beats the aggregate-profile MarkovCDF predictors for most users, the O(2) aggregate-profile MarkovCDF predictors made lower under-provision for most APs than the NG predictor.

Within each group of individual or aggregate profile MarkovCDF predictors, the higher-order predictors made lower under-provisions because they made more accurate predictions.

Figure 4.39 and Figure 4.40 show the over-provision metrics. Clearly, the NG predictor

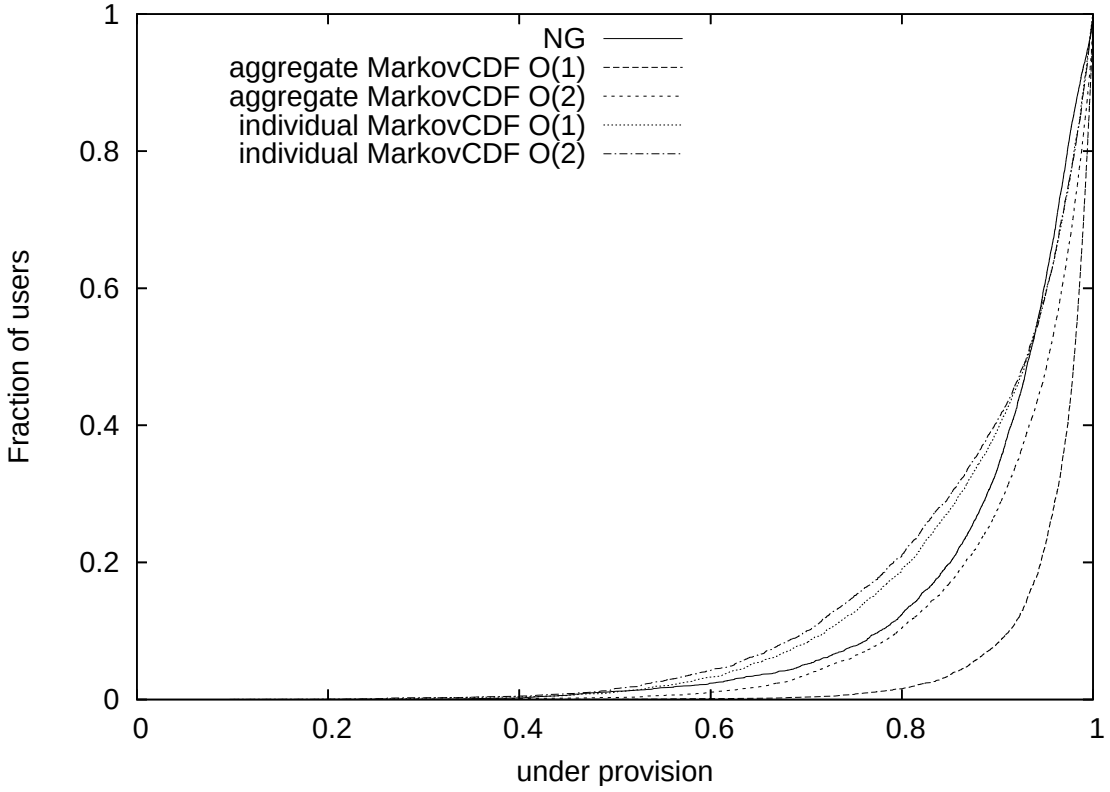


Figure 4.37: Average under-provision for users. Aggregate table and per-user table are compared. Lower under-provision numbers are better.

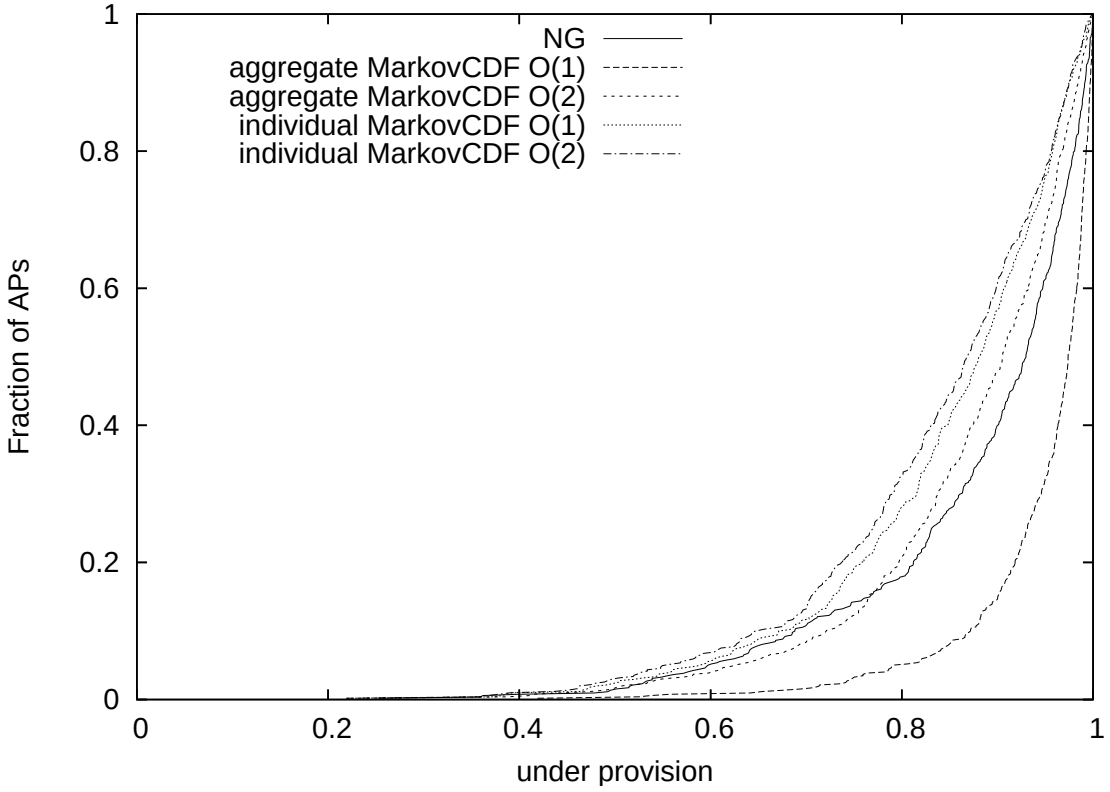


Figure 4.38: Average under-provision for APs. Aggregate table and per-user table are compared. Lower under-provision numbers are better.

far over-provided due to its assumption that the user will always leave within the period. The individual-profile MarkovCDF predictors still fared better than the aggregate profile predictors.

Both the under-provision and the over-provision were computed by accumulating the absolute errors of each individual user's predictions. For our data set, however, there were more than 2000 users during each day. With some reservation policies, an individual user's over-provision may benefit other users who are under-provided by their own prediction. To explore the overall performance of the integrated location and time predictors, we describe our VoIP case study in the next chapter.

In all four provisioning plots, for a given fraction of users, a low provision number is better. The $O(2)$ prediction was slightly better than others with the same profile. The order of predictors did not make much difference, but the use of individual versus aggregate tables made a difference. It is similar for the over-provision case in Figure 4.39 and Figure 4.40. Aggregate profile prediction made more errors in terms of the under/over-provision, because the profiles were averaged and may not reflect an individual user's mobility pattern. In Figure 4.37, half of the users had average under-provision greater than 92% for the per-user based predictors, while it was above 97% for the aggregate profile predictors. The median under-provision for APs, in Figure 4.38, was 85% and 95% for per-user based and aggregate profile predictors, respectively.

For all the predictors, when using the same order and fallback, the individual predictor beat the aggregated predictor, indicating substantial variation across users.

We include a curve for the results of the NG predictor, which did not make large under-provision errors, but it made huge over-provision errors. The NG predictor was not designed to predict the time of handoff and was thus overly aggressive at provisioning.

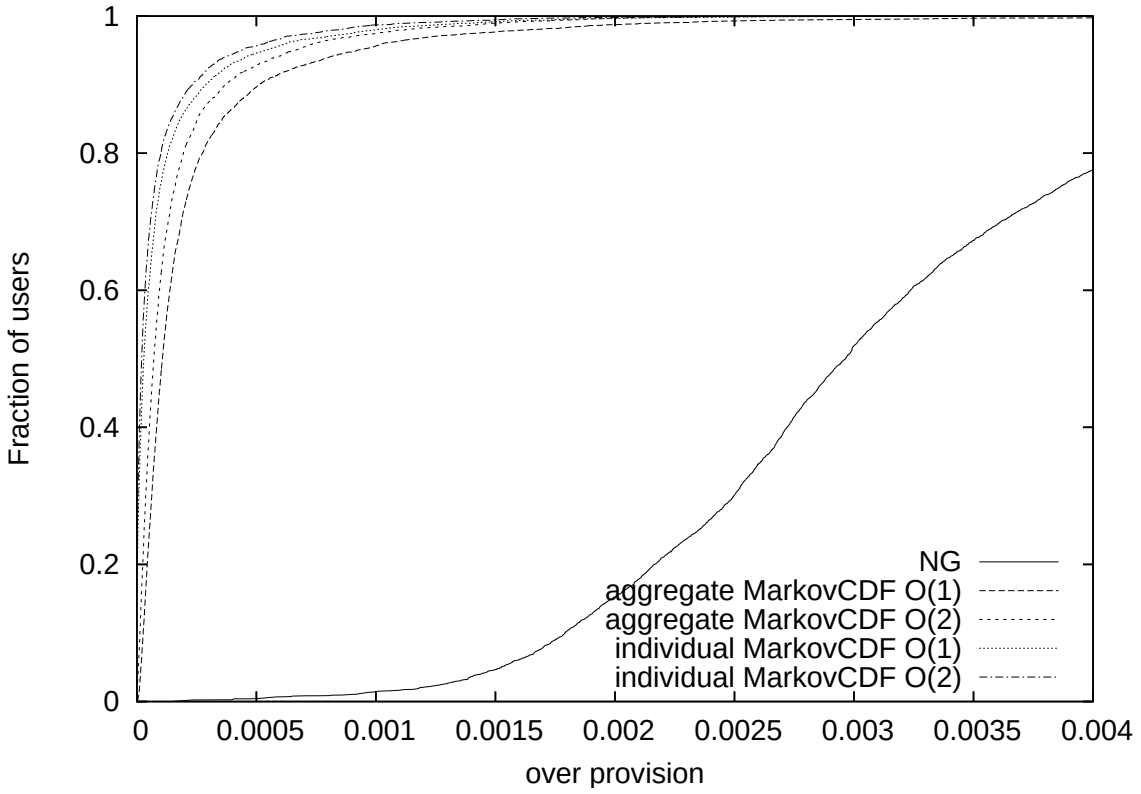


Figure 4.39: Average over provision for users. Aggregate table and per-user table are compared. Lower "over provision" numbers are better. We truncated the graph at 0.004.

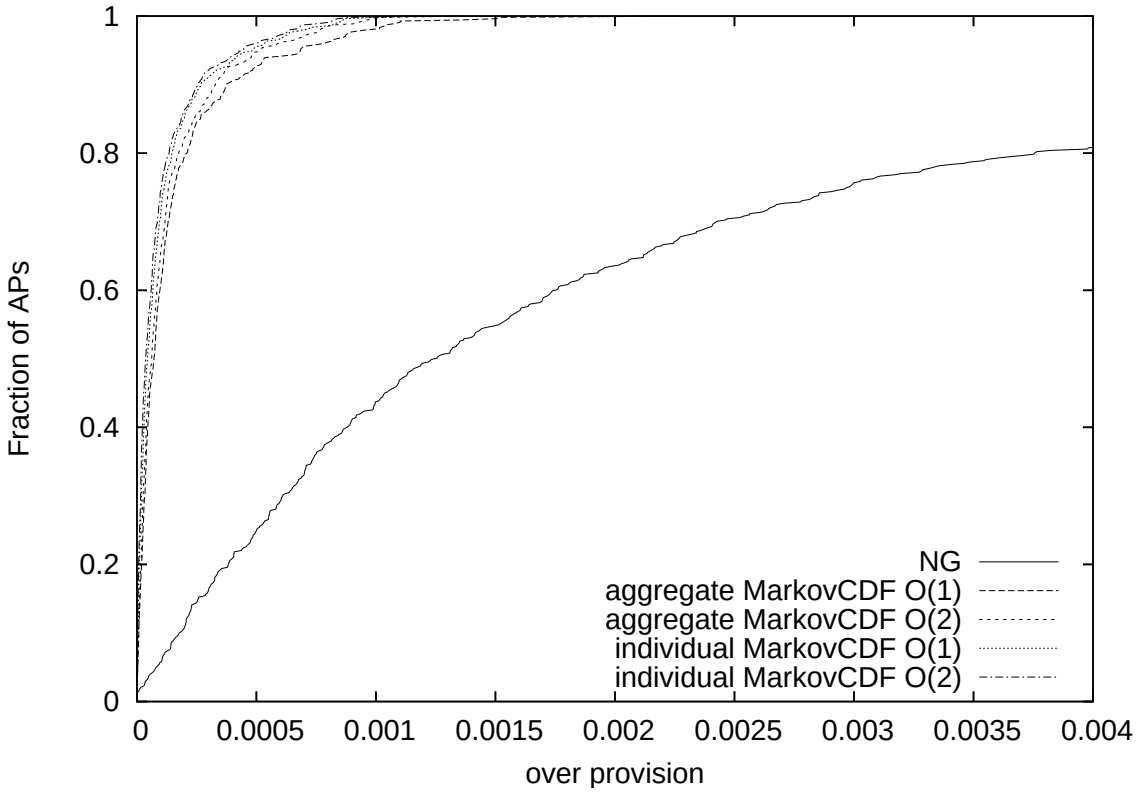


Figure 4.40: Average over provision for APs. Aggregate table and per-user table are compared. Lower "over provision" numbers are better. We truncated the graph at 0.004.

4.3 Conclusions

In this section, we summarize our evaluation of handoff prediction experiments.

4.3.1 Handoff location predictors

We conducted the first comprehensive, empirical comparison of four important classes of handoff-location predictors, to gauge which most accurately predict the next cell for a mobile wireless-network user. Specifically, we compared four major families of domain-independent on-line location predictors (Markov, LZ, PPM, and SPM) by measuring their accuracy when applied to two years of handoff traces we collected from 6,202 users of Dartmouth College's wireless network. We found, surprisingly, that the complex predictors were at best only negligibly better than the simple Markov predictor, a result that has important implications for anyone developing or using such predictors in real applications.

Many of the traces in our collection were short, fewer than 100 movements (cell changes), and all predictors fared poorly on most of those users. These predictors typically required a fair amount of history to initialize their probability tables to a useful state. In general we found that accuracy increased sharply with trace length for short traces (fewer than 100 moves), gradually for medium traces (101–1000 moves), and barely increased for long traces (over 1000 moves). Nonetheless, in this chapter we show results over all traces, sometimes focusing on long traces to highlight trends.

In general, the simple low-order Markov predictors worked as well or better than the more complex compression-based predictors, and better than high-order Markov predictors. In particular, $O(3)$ (or above) Markov predictors did not improve over $O(2)$ Markov predictors, and indeed reduced accuracy by failing to make predictions much of the time.

Most of the predictors, as defined in the literature, fail to make a prediction when faced with a context (recent history) that has never been encountered in the user's full history.

We found it was simple to add "fallback" to most predictors, allowing the predictor to use shorter and shorter context until it was able to make a prediction, and that this fallback often improved the predictor's accuracy substantially.

In particular, $O(2)$ Markov with fallback beat or equaled all of the other predictors, even though the others have better asymptotic behavior [CJv03]. We found $O(2)$ Markov with fallback to be the best overall predictor. It was simple to implement, had relatively small table size, and had the best overall accuracy.

We introduced and evaluated a simple alternative to the frequency-based approach to Markov predictors, using recency (probability 1 for the most recent transition, 0 for all others) to define the transition matrix. Although this recency approach was the best approach among $O(1)$ Markov predictors, it was the worst among $O(2)$ Markov predictors.

We found most of the literature defining these predictors to be remarkably insufficient at defining the predictors for implementation. In particular, few defined how the predictor should behave in the case of a tie, that is, when there was more than one location with the same most-likely probability. We investigated a variety of tie-breaking schemes within the Markov predictors, but found that the accuracy distribution was not sensitive to the choice.

Since some of our user traces extend over weeks or months, it is possible that the user's handoff patterns do change over time. All of our predictors assumed that the probability distributions were stable. We briefly experimented with extensions to the Markov predictors that "age" the probability tables so that more recent movements have more weight in computing probability, but the accuracy distributions did not seem significantly affected.

We examined the original LZ predictor as well as two extensions, prefix and PPM. LZ with both extensions is known as "LeZi." We found that both extensions did improve the LZ predictor's accuracy, but that the simple addition of fallback to LZ did just as well, was much simpler, and had a much smaller data structure. To be fair, LeZi tries to do more than we require, to predict the future path (not just the next move). We also examined the PPM

and SPM predictors, but found that they had little or no improvement compared to $O(2)$ Markov with fallback. We found this result to be surprising, since the LZ-based predictors and SPM have better theoretical behavior (they are asymptotically optimal for a larger class of sources). However, for our real data, with its large number of short and medium traces (as well as other phenomena that are hard to capture theoretically), we found that $O(2)$ Markov with fallback performed better in practice.

We stress that all of our conclusions are based on our observations of the predictors operating on over 6000 users, and in particular on whether a given predictor's accuracy distribution seems better than another predictor's accuracy distribution. An individual user may experience an outcome quite different from those in our conclusion. We studied the correlation between the entropy of a user's movements and the prediction accuracy. The correlation was striking, and indeed the correlation coefficient is -0.95 (a coefficient of 1.0 or -1.0 represents perfect correlation). This strong correlation indicates that some users with high entropy are doomed to poor predictability.

There was a large gap between the predictor's accuracy distribution and the "optimal" accuracy bound (a rather coarse upper bound), indicating that there is substantial room for improvement in location predictors. Our optimal bound, however, may be overly optimistic for realistic predictors, since it assumes that a predictor will predict accurately whenever the device is at a location it has visited before. We suspect that domain-specific predictors will be necessary to come anywhere close to this bound.

Overall, the best handoff-location predictors had an accuracy of about 65–72% for the median user. On the other hand, the accuracy varied widely around that median. Some applications may work well with such performance, but many applications will need more accurate predictors; we encourage further research into better predictors, as long as they are experimentally verified.

4.3.2 Joint time and location predictors

We extended our experiments of handoff-location prediction to handoff-time prediction. We explored several ways to predict of handoff time: predicting a time bucket during which the handoff would happen; predicting how long that a user would stay at the current location; and predicting the probability that a user would move to another location within a certain time period. We also defined two metrics to measure the prediction performance: one was the simple correctness measurement, the other was the earliness/lateness measurement.

We explored the predictors' dependencies across location and time. We further investigated whether group handoff or individual handoff models are more realistic. We found that the prediction quality varied widely from user to user and from access point to access point. In the over- and under-provisioning results, for example, we found the distributions to be highly skewed.

Our results show that predicting the precise time of a handoff with a granularity in the order of seconds and minutes is generally not possible by using only the handoff history of the mobile devices. Therefore, we explored predictors that can return a range of values, can express confidence in their prediction (a probability), and can express inequalities (the probability that the visit will last at least t seconds, rather than predicting departure after precisely t seconds). We observe that predictions for individual events still tend to be highly skewed even for such *soft* prediction outputs.

In summary, the contribution of this chapter is that we used extensive empirical location data collected from real users in a real network to quantitatively evaluate several handoff-location predictors, hand-off time predictors, and joint-time-and-location handoff predictors. We discovered that the more complex predictors do not necessarily perform better than the simpler Markov predictors. We further investigated the behavior of the pre-

dictors by exploring tie-breakers, alternatives such as recency versus frequency, and aging. We examined how trace length affects accuracy, and we explored how the entropy of a user's history is related to the prediction accuracy for that user. Our results bring important insights to the design of handoff predictors, and we provide valuable advice to anyone wanting to use domain-independent location predictors: keep it simple.

Chapter 5

Case Study: Bandwidth Reservation

The results of the previous section indicate that it is in general difficult to obtain accurate predictions for handoff times by relying only on the handoff history of WLAN clients. Nonetheless, even a modest prediction accuracy may help some applications that require advance information of handoffs. In this chapter, we demonstrate this point by working with a VoIP application. This application is a natural choice, considering the increasing popularity of VoIP and related multimedia applications on WLANs. Furthermore, voice has been extensively studied in the context of cellular networks, and this chapter complements those studies. Our evaluation framework can be generalized to other applications that have a notion of a session and that are also bandwidth-constrained.

Our goal is to study the effectiveness of handoff prediction on voice applications with realistic mobility traces, rather than to outperform all the prior work in cellular networks. Our evaluation simulator used the dataset [KHA04], which contains both laptops and VoIP handsets. Although most laptops were not used for voice applications, their wireless networking behavior experienced similar patterns as voice specific devices (Cisco phones and Vocera devices in our dataset), as shown in Section 2.3.

Since our prediction algorithms use historical handoffs, we assume that there is a cen-

tralized or distributed mechanism in place for (i) collecting the history, (ii) performing online predictions, and (iii) distributing prediction results to relevant application agents. The challenge of designing scalable yet optimal resource allocation that relies on online measurements is inherent to all distributed systems and is beyond the scope of this dissertation.

Although the current specification of 802.11 WLAN does not provide bandwidth reservation mechanisms, there has been much research on this issue. For example, Shankar and Choi [SC02] present a MAC-level QoS signaling protocol for IEEE 802.11e WLAN and its interaction with higher layer signaling protocols including *Resource ReSerVation Protocol* (RSVP) [BZB+97] and *Subnet Bandwidth Manager* (SBM) [GPS+00].

We assume that the wireless network is capable of supporting roaming telephone users, and that it is able to reserve bandwidth for users. When a user has an ongoing call and moves from one access point (AP) to another, we refer to that call as a *handoff call*. When a user initiates a call while associated with an AP, we refer to that call as a *new call*. All calls require a minimum dedicated bandwidth at their current AP. If the AP lacks the bandwidth for a new or handoff call, the call fails: a failed handoff call is a "call drop" and a failed new call is a "call block." The literature often assumes that call drops are much more frustrating to users than call blocks, so the goal of handoff prediction in this particular application is to reserve bandwidth, in advance of handoffs, to reduce call drops at the expense of a small increase in call blocks. Specifically, we define the drop rate

$$r_d = \frac{N_d}{N_h}, \tag{5.1}$$

where N_d is the number of dropped handoff calls, and N_h is the number of attempted handoffs.

We define the block rate

$$r_b = \frac{N_b}{N_c} \tag{5.2}$$

where N_b is the number of blocked calls, and N_c is the number of attempted calls.

Note the difference in the denominator of each equation. The drop rate r_d is not normalized by the total number of calls, or even attempted calls, but by the actual number of call handoffs, because every call handoff is an opportunity for a call drop, whereas only call attempts are opportunities for call blocks.

Our simulator was driven by the same two-month dataset we used to evaluate joint time and prediction in Section 4.2. The data were collected during October and November 2003, with 545 access points and 6,181 users. Because of the difficulty of obtaining real handoff data from wireless phone service providers, this data set is the best available data that we know to approximate wireless phone users' handoffs. Same difficulty for calling patterns, so we generated a synthetic calling pattern to model the hypothetical telephony behavior of these users.

We make the following assumptions about users and calls:

1. All calls require the same, fixed amount of bandwidth B.

2. The bandwidth of an AP is fixed and the entire capacity C of every AP is used for VoIP calls. (Or, a fixed capacity C is reserved for VoIP calls.)

3. The duration of a call is exponentially distributed with mean λ_d.

4. The length of each user's inter-call time (between the end of one call and the beginning of the next call) is exponentially distributed with mean λ_i.

5. The calling behavior of every user on the campus shares the same model and parameters.

Below, we provide the details of our bandwidth reservation policies and call admission control mechanisms.

5.1 Bandwidth reservation and admission control policies

Our predictor evaluations in Section 4.2 show that when predictors returned an handoff time as the output, the results were highly skewed, and making distinct reservations for each user became impractical. Thus, we turned our attention to predictors that can return soft values, i.e., handoff probabilities, and designed reservation mechanisms that dynamically allocate bandwidth for all handoff calls using these soft values.

We compared the simulation results when the MarkovCDF predictor was used for predictions, to the simulation results without prediction, and to the results of using all neighboring nodes as predictions, namely, the NG predictor.

We experimented with four bandwidth-reservation strategies that install bandwidth reservations on behalf of all calls in progress, using the predictors to anticipate next handoff location and time:

- **Using all proportional probabilities *non-normalized*.** For some users, for instance, when the predictor outputs a vector:

  $$\langle (A, 0.4), (B, 0.3), (C, 0.2) \rangle,$$

  we reserve $0.4B$ at access point A, $0.3B$ at access point B, and $0.2B$ at access point C, where B is the bandwidth required for a single voice call.

- **Using *normalized* proportional probabilities.** When, the predictor outputs a vector such as

  $$\langle (A, 0.4), (B, 0.3), (C, 0.2) \rangle,$$

we reserve $\frac{0.4}{0.4+0.3+0.2}B$ at access point A, $\frac{0.3}{0.4+0.3+0.2}B$ at access point B, and $\frac{0.2}{0.4+0.3+0.2}B$ at access point C, where B is the bandwidth required for a single voice call.

- **Using top-3 normalized probabilities.** When, the predictor outputs a vector such as

$$\langle (A, 0.3), (B, 0.3), (C, 0.2), (D, 0.1), (E, 0.1) \rangle,$$

we ignore all but the top three and normalize; that is, we reserve $\frac{0.3}{0.3+0.3+0.2}B$ at access point A, $\frac{0.3}{0.3+0.3+0.2}B$ at access point B, and $\frac{0.2}{0.3+0.3+0.2}B$ at access point C. We do not reserve any bandwidth at access point D or E. This method ignores the tiny probabilities (often numerous) in the prediction vector of the static neighbor predictor or aggregate CDF predictor.

- **One *B* reservation at top 3 APs.** When, the predictor returns a vector such as

$$\langle (A, 0.3), (B, 0.3), (C, 0.2), (D, 0.1), (E, 0.1) \rangle,$$

we reserve $1B$ each at access points A, B and C. As with the previous case, we ignore APs with small probabilities. This policy reserves up to $3B$, whereas the above policies reserve at most $1B$. The purpose is to compare with prior literature using this policy and to avoid under-provisioning.

For the CDF-based predictors, all these strategies update the reservations whenever a user starts a call successfully, every T seconds thereafter (where T is a design parameter) until the call ends, and whenever the call is handed off to another AP. For the static neighbor predictor, these reservations are placed as soon as a user starts a call successfully and whenever the call is handed off to another AP; the reservations are never updated as the predictions do not change over time.

A given AP may hold reservations for many callers, and it allows itself to be overbooked (that is, the total reservations may exceed the capacity of the AP). Any incoming handoff call will be accepted (not be dropped) whenever there is sufficient unused bandwidth regardless of how much bandwidth is reserved by this or any other caller; otherwise it is dropped. If the arriving caller had a reservation, its reservation is also removed. A new call will be accepted whenever there is sufficient unused *and unreserved* capacity; otherwise it is blocked. Thus, handoff calls have priority over new calls, and reservations may reduce call drops but may increase call blocks.

We summarize our admission-control procedure with pseudo-code. For each AP, let C denote the total capacity, U denote the bandwidth used by current calls, R denote the bandwidth reserved, and let R_u denote the reservation of user u at this AP, according to one of the above policies. For any AP, the reservation algorithm is as follows:

- Whenever reservations R_u are updated,

  $R = \sum_u R_u$;

- When a new call begins,

  if ($U + B \leq C - R$) // unreserved bandwidth available

  then $U = U + B$; // call allowed to begin

  else call is blocked;

- When user u with an active call hands off to this AP,

  if ($U + B \leq C$) // any bandwidth available

  then $U = U + B$; // call allowed to continue

  else call is dropped;

  $R = R - R_u$;

- When a call ends or leaves the AP,

$U = U - B$;

Reservations are cancelled ($R = R - R_u$ at all reserved APs).

5.2 Call generation

As mentioned above, we generate a synthetic calling pattern for each of our users. The duration of a call is exponentially distributed with mean λ_d. The length of each user's inter-call time (between the end of one call and the beginning of the next call for the same user) is exponentially distributed with mean λ_i.

The Dartmouth network is not yet heavily loaded. To increase congestion, we set both the mean duration λ_d and the mean inter-call time λ_i to 15 minutes (900 seconds).

5.3 Training

The quality of predictions in the CDF predictors depends on whether, in the past, there have been enough observations at the APs where the user will be associated. To prevent such lack of observation, we train the predictor with the handoff traces of the first month without making calls or reservations, and start measurements in the second month. Thus, when training is applied, we measure the application performance during the second month only. For the sake of completeness, we also provide the performance results with no predictor training.

Table 5.1: Predictor usage

predictor		Markov	MA	CDF	MarkovCDF	NG
usage	location	yes	no	no	no	no
	time of day	yes	no	no	no	no
	duration	yes	yes	yes	no	no
	integrated	no	no	no	yes	no
metrics	accuracy	yes	yes	yes	no	no
	earliness/lateness	yes	yes	yes	no	no
	under/over provision	no	no	no	yes	yes
dependency	location†	yes/no	yes/no	yes	yes	yes
	time‡	no	no	yes	yes	no
profile	individual	yes	yes	yes	yes	no
	aggregate	yes	no	yes	yes	yes

† The prediction depends on current and/or more previous visited locations.
‡ The prediction depends on how long a user has been at the current location.

5.4 Results

We developed a simulator[1] that was driven by the handoff traces collected at Dartmouth and the calls generated using the pattern described in Section 5.2.

In Chapter 3, we present a set of predictors. Now we choose the applicable predictors for our simulator. Table 5.1 provides a summary of the usage, metric, dependency, and profile for those predictors.

Since MarkovCDF and NG can be measured in under/over provision, they are suitable for bandwidth reservations. We ran our simulation using the following predictors:

- MarkovCDF individual predictor $O(2)$ with fallback,
- MarkovCDF aggregate predictor $O(2)$ with fallback,
- NG predictor trained on the October 2003 data.

The two MarkovCDF predictors were run with and without training. All three predictors were run with the four reservation strategies described above.

[1]Thanks to Udayan Deshpande for implementing and experimenting with the simulation.

We ran each case 10 times, each time with a different seed for the traffic-generation module. Unless otherwise mentioned, the AP capacity $C = 5B$ and T is 300 seconds. We plot the relative improvement in the drop rate (Figure 5.1) and relative worsening of the block rate (Figure 5.2) as compared with a system without prediction-based bandwidth reservation. Note that these plots show the *means of the ratios*, with mean computed across the many seeds; that is, we compute for each predictor

$$\text{mean}\left(\frac{r'_d}{r_d}\right)$$

and

$$\text{mean}\left(\frac{r_b}{r'_b}\right),$$

where r'_d is the base drop rate, i.e., when no predictor was used, and r_d is the drop rate after using a given predictor. Similarly, r'_b is the block rate of using no predictor and r_b is the block rate after using a predictor.

The reason we use improvements and worsenings in the plots rather than the nominal values is that the drop rates and the block rates vary greatly over the different seeds to the call generation module. Hence it makes more sense to summarize the *improvements* in drop rates and the *worsenings* in block rates. To read the plots, if the value for the "Base *DR*/Reserved *DR*" ratio was 2, we say that the drop rate without prediction was double the drop rate with prediction. Similarly, if the value for the "Reserved *BR*/Base *BR*" ratio was 1.5, we say that the block rate with prediction was 50% higher than the block rate without prediction.

In Figures 5.1 and 5.3 we observe that the Base *DR*/Reserved *DR* ratio lies between 1.4 and 22.4. From Table 5.2 we can see that the average *DR* with no prediction was 13.8%;

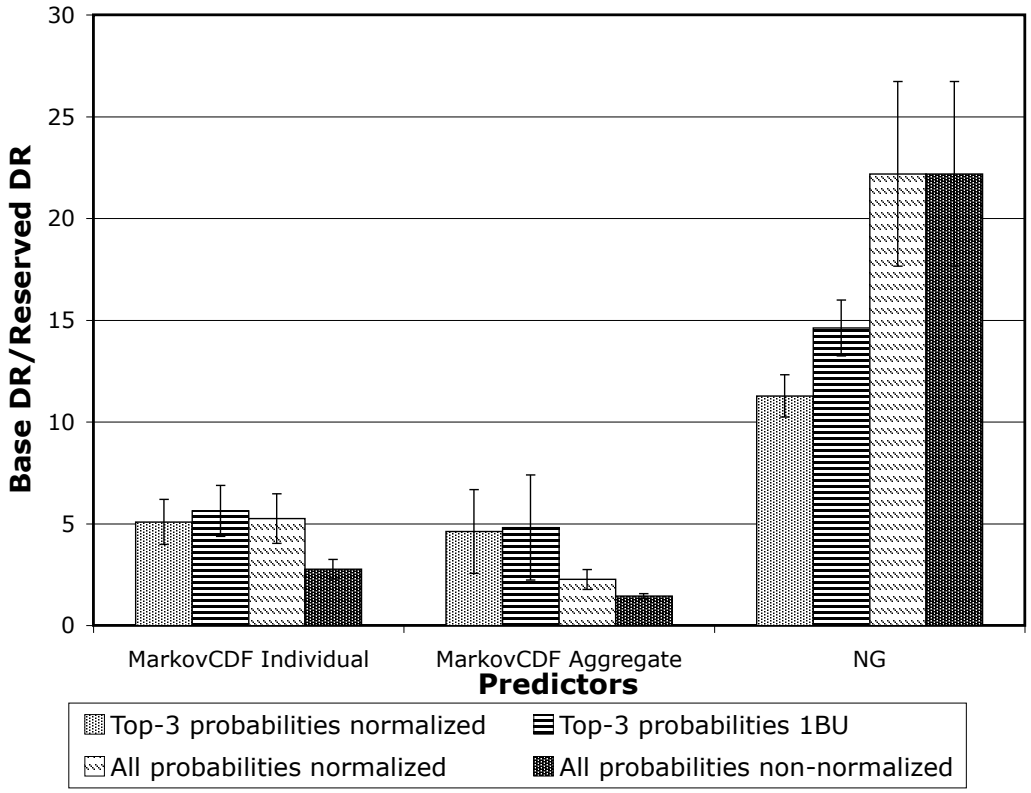


Figure 5.1: The ratio (Base *DR*/Reserved *DR*) for all the predictors with training. The error bars show the standard deviation σ among the various seeds of the simulation. A higher ratio is better.

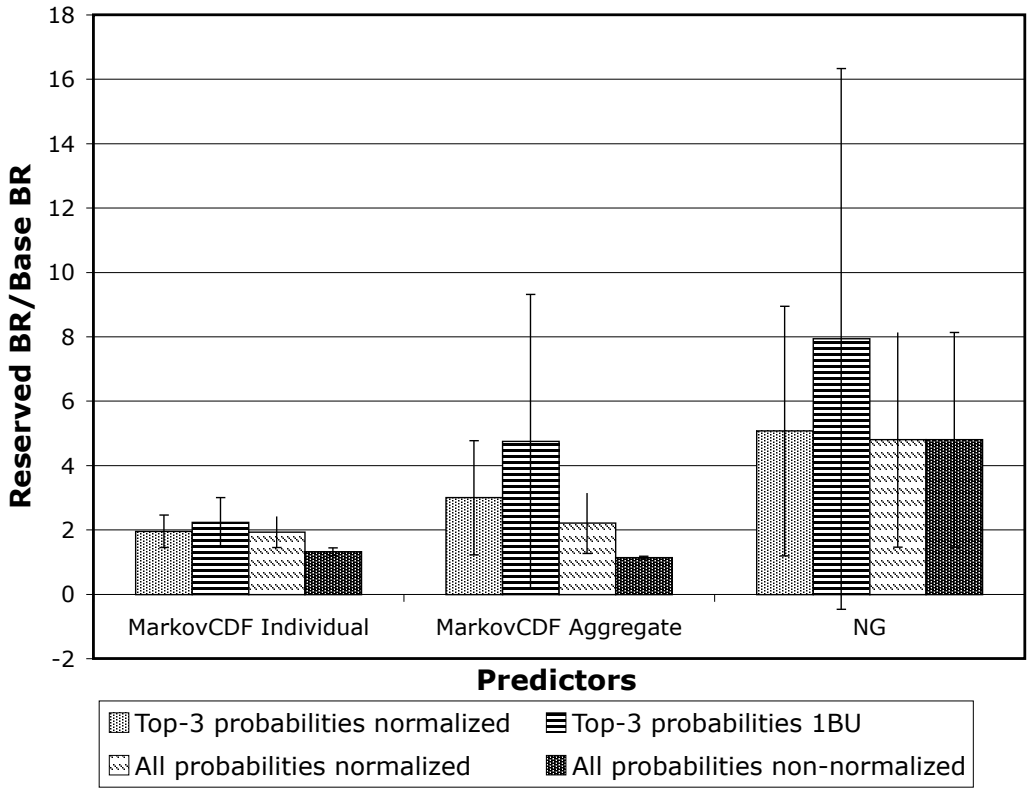


Figure 5.2: The ratio (Reserved *BR*/Base *BR*) for all the predictors with training. The error bars show the standard deviation σ among the various seeds of the simulation. A higher ratio is worse.

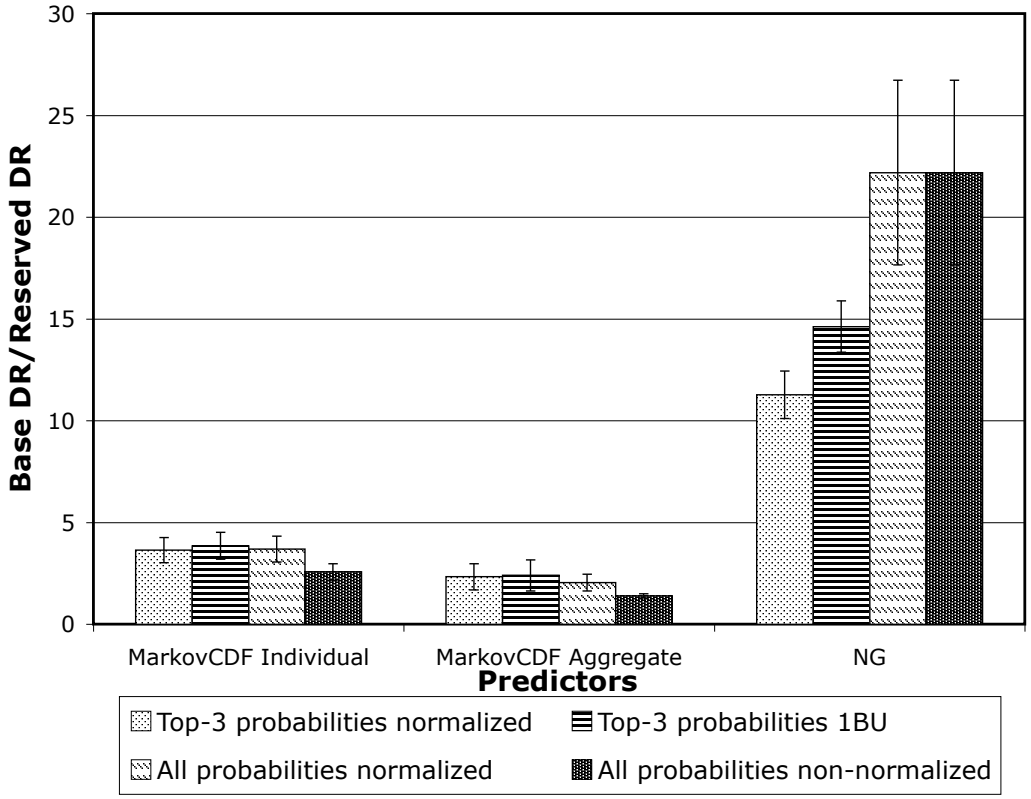


Figure 5.3: The ratio (Base *DR*/Reserved *DR*) for all the predictors without training. The error bars show the standard deviation σ among the various seeds of the simulation. A higher ratio is better.

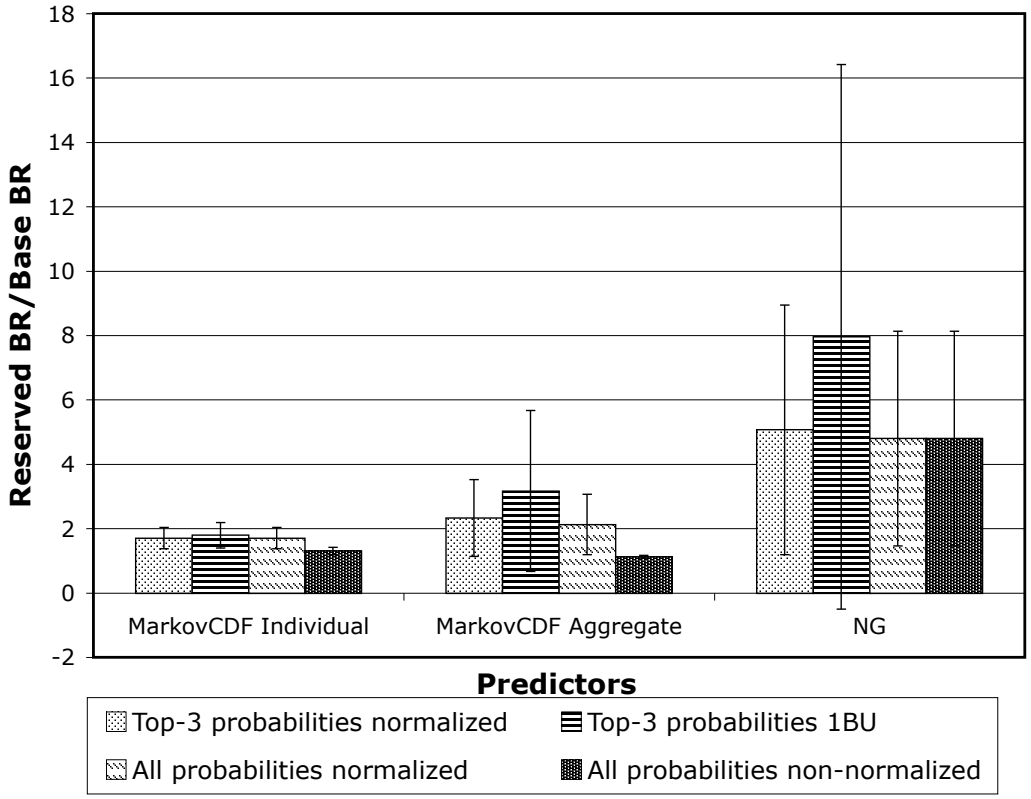


Figure 5.4: The ratio (Reserved *BR*/Base *BR*) for all the predictors without training. The error bars show the standard deviation σ among the various seeds of the simulation. A higher ratio is worse.

Table 5.2: Absolute values of call drop rate and block rate with training

	Scheme	No Prediction		Individual		Aggregate		NG	
		Average†	σ‡	Average	σ	Average	σ	Average	σ
drop-rate r_d	All norm	13.80	7.58	2.98	2.23	7.01	4.73	0.69	0.51
	All	13.80	7.58	5.57	3.86	10.04	6.03	0.69	0.51
	Top 3	13.80	7.58	2.74	2.01	4.36	3.53	0.96	0.58
	3 norm	13.80	7.58	3.03	2.23	4.28	3.48	1.24	0.74
block-rate r_b	All norm	11.74	10.35	18.72	14.1	18.84	12.98	33.23	19.31
	All	11.74	10.35	14.51	12.13	12.88	11.05	33.23	19.31
	Top 3 full	11.74	10.35	20.24	14.50	26.81	13.51	40.61	17.66
	3 norm	11.74	10.35	18.85	14.14	22.56	14.15	33.48	18.94

†The values in the *average* column are means of the percentage rates over 10 seeds in the simulation.
‡The σ value is the standard deviation over these seeds.

prediction-based bandwidth reservation reduced drop rate to between 0.7% and 10%. We see that the individual predictors tended to perform better than the aggregate predictors in terms of drop rate and block rate. This implies that an individual user was more likely to repeat her own patterns than to follow others' patterns at the same AP.

Intuitively, if the drop rate improves, the block rate should worsen and vice versa. See Figure 5.2 and 5.4 for plots of the lock rate changes. In an extreme case, the entire bandwidth would be reserved for handoff calls and all new calls would be blocked. We expect that the choice of predictor depends on what block rate one wishes to tolerate. From Table 5.2, we see that on average the block rate with no prediction was 11.7%. With reservation, the block rate increases to between 13% and 41%. The 40% block rate was observed in the case of the NG predictor. It is clearly important to reserve bandwidth, but not too much. The ideal ratio may be achieved by the right choice of predictor and by tuning parameters such as T.

The expectation that improvements in drop rate are correlated to the worsening in the block rate turn out to be false in some predictors. For example, in case of the MarkovCDF individual predictor, when compared to the MarkovCDF aggregate predictor (Figure 5.1), we expect the block rate for the former to worsen more than that for the latter predictor; however, the opposite is true (Figure 5.2). We attribute this effect to higher probabilities being returned by the MarkovCDF individual predictor for the more likely APs than the MarkovCDF aggregate predictor. In effect, the predictions for the MarkovCDF individual are *better*. Clearly, if smart predictions are made, we should be able to maintain a lower block rate in spite of improvements in the drop rate.

Note that the standard deviation in the worsening of the block rate is very high. However, the standard deviation for the improvement in drop rate is quite low. Hence we submit that, although the drop rate was controllable using the reservations, it was not tightly coupled with the block rate.

In the trained data set (Figure 5.1 and Figure 5.2), we note that the MarkovCDF individual predictor improved the drop rate more than the MarkovCDF aggregate predictor. At the same time they worsen the block rate to a lesser extent. This difference occurred because users on the campus network had individual patterns that show up in handoff traces that are sufficiently long, but every user may not have behaved the same way and hence the aggregate predictors predicted less accurately.

As expected, the improvement in drop rate is higher in the results of the simulations that used predictors with one month of training data (Figure 5.1) than in the results of the simulations without training (Figure 5.3). In the case of the static neighbor predictor, we used training data from the month of October 2003. The NG predictor, however, does not learn while it is used for predictions during simulation. Hence we use the same figures for the NG in both figures.

The worsening in block rate is also higher in the results of the simulations that used predictors with one month of training data (Figure 5.2) than in the results of the simulations without training (Figure 5.4). The reason for this result is because, upon learning for the extra month, the predictor predicts a wider range of future APs where reservations are made, hence blocking other calls that may arrive at these APs.

5.5 Conclusion

In this chapter, we focus on quantifying the quality of mobile handoff predictions for VoIP applications in a real large-scale WLAN environment.

We evaluated our MarkovCDF predictors (individual and aggregated) using two months' handoff traces for a VoIP application with four bandwidth reservation schemes: top-3 probabilities normalized, top-3 probability 1 bandwidth unit, all probabilities normalized, and all probabilities non-normalized.

The "top-3 probabilities normalized" reservation scheme did not provide much benefit for the MarkovCDF individual predictor in terms of block rate. The reason is that there were often not many more than 3 APs in the prediction vector of the MarkovCDF individual predictor anyway. So the limit of 3 that the "top-3 probabilities" scheme imposed is superfluous.

The MarkovCDF aggregate predictors benefited from the "top-3 probabilities normalized" scheme in terms of drop rate, because concentrating of reservations at fewer APs provided more resources for users on the network that have similar behavior. Due to these large reservations, however, there was more blocking and hence a higher block rate.

The neighbor graph predictor performed worse using the "top-3 probabilities normalized" reservation scheme in both metrics. The drop rate did not improve as much as the "all probabilities" scheme because the "top-3" scheme made fewer reservations. The block rate worsened slightly more because of the concentrated reservations, similar to the MarkovCDF aggregate predictor.

Hence it is a good idea to use the "top-3 probabilities normalized" reservation scheme, as it cut down on the operations to be performed (such as reservations, notification of the APs). Even in the NG predictor, it was beneficial to use this scheme; despite the higher drop rate, the increase in block rate was far too large when there were many neighbors. There is a tradeoff between the number of reservations, and the drop rate and the block rate.

In our simulation, the "top-3 probabilities $1B$" scheme provides only slight benefit in drop rate over the others and worsens the block rate considerably more. The reason is that it reserves $3B$ per prediction rather than $1B$, system wide. Hence any scheme with proportional probabilities is recommended.

For the two MarkovCDF predictors, the normalized version of the "all probabilities" reservation scheme improved the drop rate more than the non-normalized version at the

cost of higher block rate. Clearly the reason is that larger reservations were made on behalf of the user.

When comparing the simulations with the trained predictors against the simulations with the untrained predictors, we recall that the neighbor predictor is trained with the first month's data regardless, but it did not learn during the second month like the MarkovCDF predictors do. As expected, the training in the MarkovCDF predictors generally benefits the drop rate while worsening the block rate to a lesser extent.

These results show that even with moderate prediction accuracy, we can improve the drop rate, while not worsening the block rate as much. We see that the simplistic neighbor graph predictor will block considerably more calls than is acceptable in most systems, because it does not predict time at all, neither does it focus on location prediction. Using the predictors that we have presented helps us to be more discriminating in our reservations. Ultimately, the choice of the reservation schemes with the predictors is important, and the selection of a particular policy hinges on an operator's relative importance of reducing dropped calls versus avoiding more blocked calls.

Chapter 6

Routing in Mobile Opportunistic Networks

So far we have studied the handoff-prediction techniques and their applications in wireless infrastructure networks. We now consider a different sort of network.

Routing in mobile ad hoc networks have been studied extensively. Most of these studies assume that a contemporaneous end-to-end path exists for the network nodes. Some mobile ad hoc networks, however, may not satisfy this assumption. In mobile sensor networks [WW06], sensor nodes may turn off to preserve power. In wild-animal tracking networks [JOW+02], animals may roam far away from each other. Other networks, such as "pocket-switched" networks [HCS+05], battlefield networks [LR03, HA06] and transportation networks [BGJL06, LCGZ05], may experience similar disconnections due to mobility, node failure, or power-saving efforts.

One solution for message delivery in such networks is that the source passively waits for the destination to be in the communication range and then delivers the message. Another active solution is to flood the message to any nodes in communication range. The receiving nodes carry the message and repeatedly flood the network with the message. Both solutions

have obvious advantages and disadvantages: the first may have low delivery ratio while using few resources; the second may have high delivery ratio while using many resources.

Many other opportunistic routing protocols have been proposed in the literature. Few of them, however, were evaluated in realistic network settings, or even in realistic simulations, due to the lack of any realistic people mobility model. Random walk or random way-point mobility models are often used to evaluate the performance of those routing protocols. Although these synthetic mobility models have received extensive interest by mobile ad hoc network researchers [CBD02], they do not reflect people's mobility patterns [JLB05]. Realising the limitations of using random mobility models in simulations, a few researchers have studied routing protocols in mobile opportunistic networks with realistic mobility traces. In the Haggle project, Chaintreau et al. [CHC+06] theoretically analyzed the impact of routing algorithms over a model derived from a realistic mobility data set. Su et al. [SGdL06] simulated a set of routing protocols in a small experimental network. Those studies help researchers better understand the theoretical limits of opportunistic networks, and the routing protocol performance in a small network (20–30 nodes).

Deploying and experimenting large-scale mobile opportunistic networks is difficult, so we, too, resort to simulation. Instead of using a complex mobility model to mimic people's mobility patterns, we used mobility traces collected in a production wireless network at Dartmouth College to drive our simulation. Our message-generation model, however, was synthetic.

We study protocols for routing messages between wireless networking devices carried by people. We assume that people send messages to other people occasionally, using their devices; when no direct link exists between the source and the destination of the message, other nodes may relay the message to the destination. Each device represents a unique person (it is out of the scope of our work when a device maybe carried by multiple people). Each message is destined for a specific person and thus for a specific node carried by that

person. Although one person may carry multiple devices, we assume that the sender knows which device is the best to receive the message. We do not consider multicast or geocast in this dissertation.

Using realistic contact traces, which were derived from Dartmouth College's dataset as described in Section 2.1, we evaluated the performance of three "naive" routing protocols (direct-delivery, epidemic, and random) and two prediction-based routing protocols, PRoPHET [LDS04] and Link-State [SGdL06]. We also propose a new prediction-based routing protocol, and compare it to the above in our evaluation.

6.1 Routing Protocol

A routing protocol is designed for forwarding messages from one node (source) to another node (destination). Any node may generate messages for any other node, and may carry messages destined for other nodes. We consider only messages that are unicast (single destination).

DTN routing protocols can be described in part by their *transfer probability* and *replication probability*; that is, when one node meets another node, what is the probability that a message should be transfered and if so, whether the sender should retain its copy. Two extremes are the direct-delivery protocol and the epidemic protocol. The former transfers with probability 1 when the node meets the destination, 0 for others, and no replication. The latter uses transfer probability 1 for all nodes and unlimited replication. Both these protocols have their advantages and disadvantages. All other protocols are between the two extremes.

First, we define the notion of contact between two nodes. Then we describe five existing protocols before presenting our own proposal.

A *contact* is defined as a period of time during which two nodes have the opportunity

to communicate. Although we are aware that wireless technologies differ, we assume that a node can reliably detect the beginning and end time of a contact with nearby nodes. A node may be in contact with several other nodes at the same time.

The contact history of a node is a sequence of contacts with other nodes. Node i has a contact history H_{ij}, for each other node j, which denotes the historical contacts between node i and node j. We record the start and end time for each contact; however, the last contacts in the node's contact history may not have ended.

6.1.1 Direct Delivery Protocol

In this simple protocol, a message is transmitted only when the source node can directly communicate with the destination node of the message. In mobile opportunistic networks, however, the probability for the sender to meet the destination may be low, or even zero.

6.1.2 Epidemic Routing Protocol

The epidemic routing protocol [VB00] floods messages into the network. The source node sends a copy of the message to every node that it meets. The nodes that receive a copy of the message also send a copy of the message to every node that they meet. Eventually, a copy of the message arrives at the destination of the message.

This protocol is simple, but may use significant resources; excessive communication may drain each node's battery quickly. Moreover, since each node keeps a copy of each message, storage is not used efficiently, and the capacity of the network is limited.

At a minimum, each node must expire messages after some amount of time or stop forwarding them after a certain number of hops. After a message expires, the message will not be transmitted and will be deleted from the storage of any node that holds the message.

An optimization to reduce the communication cost is to transfer *index messages* before

transferring any data message. The index messages contain IDs of messages that a node currently holds. Thus, by examining the index messages, a node only transfers messages that are not yet contained on the other nodes.

6.1.3 Random Routing

An obvious approach between the above two extremes is to select a transfer probability between 0 and 1 to forward messages at each contact. The replication factor can also be a number between 0 (none) and 1 (all). We use a simple replication strategy that makes no replicas. The message is transfered every time the transfer probability is greater than the threshold. The message has some chance of being transferred to a highly mobile node, and thus may have a better chance to reach its destination before message expires.

6.1.4 PRoPHET Protocol

PRoPHET [LDS04] is a Probabilistic Routing Protocol using History of past Encounters and Transitivity to estimate each node's delivery probability for each other node. When node i meets node j, the delivery probability of node i for j is updated by

$$p'_{ij} = (1 - p_{ij})p_0 + p_{ij}, \tag{6.1}$$

where p_0 is an initial probability, a design parameter for a given network. Lindgren et al. [LDS04] chose 0.75, as did we in our evaluation. When node i does not meet j for some time, the delivery probability decreases by

$$p'_{ij} = \alpha^k p_{ij}, \tag{6.2}$$

where α is the aging factor $(\alpha < 1)$, and k is the number of time units since the last update.

The PRoPHET protocol exchanges index messages as well as delivery probabilities. When node i receives node j's delivery probabilities, node i may compute the transitive delivery probability through j to z with

$$p'_{iz} = p_{iz} + (1 - p_{iz})p_{ij}p_{jz}\beta, \tag{6.3}$$

where β is a design parameter for the impact of transitivity; we use $\beta = 0.25$ (as in [LDS04]).

ZebraNet [JOW+02] used a simple history-based routing protocol, which uses a similar equation as 6.2 to estimate the probability that a node can communicate with the destination, the base station. Only one base station exists in ZebraNet.

6.1.5 Link-State Protocol

Su et al. [SGdL06] use a link-state approach to estimate the weight of each path from the source of a message to the destination. They use the median inter-contact duration or exponentially aged inter-contact duration as the weight on links. The exponentially aged inter-contact duration of node i and j is computed by

$$w'_{ij} = \alpha w_{ij} + (1 - \alpha)(t - t_{ij}), \tag{6.4}$$

where t is the current time, t_{ij} is the time of last contact, and α is the aging factor. At the first contact, we just record the contact time.

Nodes share their link-state weights when they can communicate with each other, and messages are forwarded to the node that have the path with the lowest link-state weight.

6.2 Timely-Contact Probability

We, too, use historical contact information to estimate the probability of meeting other nodes in the future. But our method differs in that we estimate the contact probability within a period of time. For example, what is the contact probability in the next hour? Neither PRoPHET nor Link-State considers time in this way.

One way to estimate the "timely-contact probability" is to use the ratio of the total contact duration to the total time. However, this approach does not capture the frequency of contacts. For example, one node may have a long contact with another node, followed by a long non-contact period. A third node may have a short contact with the first node, followed by a short non-contact period. Using the above estimation approach, both examples would have similar contact probability. In the second example, however, the two nodes have more frequent contacts.

We design a method to capture the contact frequency of mobile nodes. For this purpose, we assume that even short contacts are sufficient to exchange messages.[1]

The probability for node i to meet node j is computed by the following procedure. We divide the contact history H_{ij} into a sequence of n periods of ΔT starting from the start time (t_0) of the first contact in history H_{ij} to the current time. We number each of the n periods from 0 to $n-1$, then check each period. If node i had any contact with node j during a given period m, which is $[t_0 + m\Delta T, t_0 + (m+1)\Delta T)$, we set the contact status I_m to be 1; otherwise, the contact status I_m is 0. The probability $p_{ij}^{(0)}$ that node i meets node j in the next ΔT can be estimated as the average of the contact status in prior intervals:

$$p_{ij}^{(0)} = \frac{1}{n} \sum_{m=0}^{n-1} I_m. \tag{6.5}$$

[1] In our simulation, however, we accurately model the communication costs and some short contacts will not succeed in transfer of all messages.

To adapt to the change of contact patterns, and reduce the storage space for contact histories, a node may discard old history contacts; in this situation, the estimate would be based on only the retained history.

The above probability is the direct contact probability of two nodes. When considering only one-node forwarding, we chose the node that has the highest contact probability to forward to. This method is like the $O(0)$ Markov predictor, which picks the most frequent symbol. We are also interested in the probability that we may be able to pass a message through a sequence of k nodes. Therefore, we need not only the node with highest contact probability, but also all other nodes. With all nodes' contact probability, we can compute the k-order probability. Assuming nodes' contact events are independent, we define the k-order probability inductively,

$$p_{ij}^{(k)} = p_{ij}^{(0)} + \sum_{r} p_{ir}^{(0)} p_{rj}^{(k-1)}, \tag{6.6}$$

where r is any node other than i or j. Since node contact events may not be entirely independent, we recognize that the above equation only estimates an *approximate* probability. Indeed, all of the prediction-based methods use heuristic approximations and incomplete data, given the nature of this routing problem.

6.3 Our Routing Protocol

We first consider the case of a two-hop path, that is, with only one relay node. We consider two approaches: either the receiving neighbor decides whether to act as a relay, or the source decides which neighbors to use as relay.

6.3.1 Receiver Decision

Whenever a node meets other nodes, they exchange all their messages (or as above, index messages). If the destination of a message is the receiver itself, the message is delivered. Otherwise, if the probability of delivering the message to its destination through this receiver node within ΔT is greater than or equal to a certain threshold, the message is stored in the receiver's storage to forward to the destination. If the probability is less than the threshold, the receiver discards the message. Notice that our protocol replicates the message whenever a good relay comes along.

6.3.2 Sender Decision

To make decisions, a sender must have the information about its neighbors' contact probability with a message's destination. Therefore, meta-data exchange is necessary.

When two nodes meet, they exchange a meta-message, containing an unordered list of node IDs for which the sender of the meta-message has a contact probability greater than the threshold.

After receiving a meta-message, a node checks whether it has any message that is destined to its neighbor or to a node in the node list of the neighbor's meta-message. If it has, it sends a copy of the message.

When a node receives a message, if the destination of the message is the receiver itself, the message is delivered. Otherwise, the message is stored in the receiver's storage for forwarding to the destination.

6.3.3 Multi-node Relay

When we use more than two hops to relay a message, each node needs to know the contact probabilities along all possible paths to the message destination.

Every node keeps a contact probability matrix, in which each cell p_{ij} is a contact probability between to nodes i and j. Each node i computes its own contact probabilities (row i) with other nodes using Equation (6.5) whenever the node ends a contact with other nodes. Each row of the contact probability matrix has a version number; the version number for row i is increased only when node i updates the matrix entries in row i. Other matrix entries are updated through exchange with other nodes when they meet.

When two nodes i and j meet, they first exchange their contact probability matrices. Node i compares its own contact matrix with node j's matrix. If node j's matrix has a row l with a higher version number, then node i replaces its own row l with node j's row l. Likewise node j updates its matrix. After the exchange, the two nodes will have identical contact probability matrices.

Next, if a node has a message to forward, the node estimates its neighboring node's order-k contact probability to contact the destination of the message using Equation (6.6). The order k is a design factor for the multi-node relay protocols. If $p_{ij}^{(k)}$ is above a threshold, or if j is the destination of the message, node i will send a copy of the message to node j.

All the above effort serves to determine the transfer probability when two nodes meet. The replication decision is orthogonal to the transfer decision. In our implementation, we always replicate. Although PRoPHET [LDS04] and Link-State [SGdL06] do no replication, as described, we added replication to both protocols for better comparison to our protocol.

6.4 Evaluation Results

We evaluate and compare the results of direct delivery, epidemic, random, PRoPHET, Link-State, and timely-contact routing protocols.

6.4.1 Mobility traces

We use real mobility data collected at Dartmouth College. Dartmouth College has collected association and disassociation messages from devices on its wireless network since spring 2001 [KHA04]. Each message records the wireless card MAC address, the time of association/disassociation, and the name of the access point. We treat each unique MAC address as a node. For more information about Dartmouth's network and the data collection, see previous studies [HKA04, KE05].

Our data are not contacts in a mobile ad hoc network. We can approximate contact traces by assuming that two users can communicate with each other whenever they are associated with the same access point. Chaintreau et al. [CHC+06] used Dartmouth data traces and made the same assumption to theoretically analyze the impact of human mobility on opportunistic forwarding algorithms. This assumption may not be accurate,[2] but it is a good first approximation. In our simulation, we imagine the same clients and same mobility in a network with no access points. Since our campus has full Wi-Fi coverage, we assume that the location of access points had little impact on users' mobility.

We simulated one full month of trace data (November 2003), with 5,142 users. Although prediction-based protocols require prior contact history to estimate each node's delivery probability, our preliminary results show that the performance improvement of warming-up over one month of trace was marginal. Therefore, for simplicity, we show the results of all protocols without warmup.

[2]Two nodes may not have been able to directly communicate while they were at far sides of an access point, or two nodes may have been able to directly communicate if they were between two adjacent access points.

6.4.2 Simulator

We developed a custom simulator.[3] Since we used contact traces derived from real mobility data, we did not need a mobility model and omitted physical and link-layer details for node discovery. We were aware that the time for neighbor discovery in different wireless technologies varies from less than one second to several seconds. Furthermore, connection establishment also takes time, such as DHCP. In our simulation, we assumed that the nodes could discover and connect to each other instantly when they were associated with the same AP. To accurately model communication costs, however, we simulated some MAC-layer behaviors, such as collision.

The default settings of the network of our simulator are listed in Table 6.1, using the values recommended by other papers [SGdL06, LDS04]. The message probability was the probability of generating messages, as described below in Section 6.4.3. The default transmission bandwidth was 11 Mb/s. When one node tried to transmit a message, it first checked whether another node was transmitting. If it was, the node backed off for a random number of slots. Each slot was 1 millisecond, and the maximum number of backoff slots was 30. The size of messages was uniformly distributed between 80 bytes and 1024 bytes. The hop count limit (HCL) was the maximum number of hops before a message should stop forwarding. The time to live (TTL) was the maximum duration that a message may exist before expiring. The storage capacity was the maximum space that a node can use for storing messages. For our routing method, we used a default prediction window ΔT of 10 hours and a probability threshold of 0.01. The replication factor r was not limited by default, so the source of a message transferred the messages to any other node that had a contact probability with the message destination higher than the probability threshold.

[3] We tried to use a general network simulator (ns2), which was extremely slow when simulating a large number of mobile nodes (in our case, more than 5000 nodes), and provided unnecessary detail in modeling lower-level network protocols.

Table 6.1: Default settings of the opportunistic network simulation

Parameter	Default value
message probability	0.001
bandwidth	11 Mb/s
transmission slot	1 millisecond
max backoff slots	30
message size	80–1024 bytes
hop count limit (HCL)	unlimited
time to live (TTL)	unlimited
storage capacity	unlimited
prediction window ΔT	10 hours
probability threshold	0.01
contact history length	20
replication	always
aging factor α	0.9 (0.98 PRoPHET)
initial probability p_0	0.75 (PRoPHET)
transitivity impact β	0.25 (PRoPHET)

6.4.3 Message generation

After each contact event in the contact trace, we generated a message with a given probability; we chose a source node and a destination node randomly using a uniform distribution across nodes seen in the contact trace up to the current time. When there were more contacts during a certain period, there was a higher likelihood that a new message was generated in that period. This correlation is not unreasonable, since there were more movements and contacts during the day than during the night. Figure 6.1 shows the statistics of the numbers of movements and the numbers of contacts during each hour. These statistics were accumulated across all users through the whole month. The plot shows a clear diurnal activity pattern; the activities were lowest around 5am and peaked between 4pm and 5pm. We assume that in some applications, network traffic exhibits similar patterns; that is, people send more messages during the day, too.

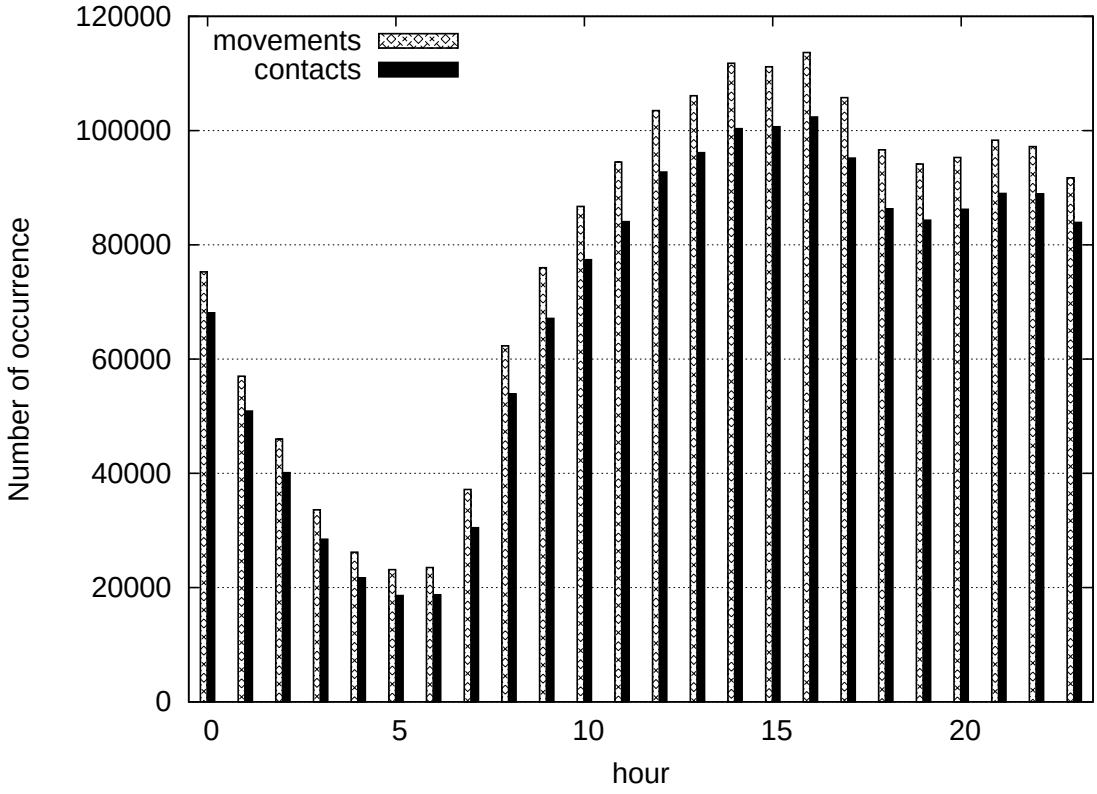


Figure 6.1: Movements and contacts during each hour

6.4.4 Metrics

We define a set of metrics that we use in evaluating routing protocols in opportunistic networks:

- *delivery ratio*, the ratio of the number of messages delivered to the number of total messages generated.
- *delay*, the duration between a message's generation time and the message's delivery time.
- *message transmissions*, the total number of messages transmitted during the simulation across all nodes.
- *meta-data transmissions*, the total number of meta-data units transmitted during the simulation across all nodes.
- *message duplications*, the number of times a message copy occurred.
- *storage usage*, the max and mean of maximum storage (bytes) used across all nodes.

6.4.5 Results

Here we compare our simulation results of the six routing protocols.

Figure 6.2 shows the delivery ratio of all the protocols, with different TTLs. (In all the plots in this section, "prediction" stands for our method, "state" stands for the Link-State protocol, and "prophet" represents PRoPHET.) Although we had 5,142 users in the network, the direct-delivery and random protocols had low delivery ratios (note the log scale). Even for messages with an unlimited lifetime, only 59 out of 2077 messages were delivered during this one-month simulation. The delivery ratio of epidemic routing was the best. The three prediction-based routing schemes had low delivery ratios, compared with epidemic

routing. Although our method was slightly better than the other two, the advantage was marginal. Note that with a 10-hour TTL, the three prediction-based routing protocols had only about 4% messages delivered. This low delivery ratio limits the applicability of these routing protocols in practice.

The high delivery ratio of epidemic routing came with a price: excessive transmissions. Figure 6.3 shows the number of message data transmissions. The number of message transmissions in epidemic routing was more than 10 times higher than for the prediction-based routing protocols. Obviously, the direct delivery protocol had the lowest number of message transmissions — the number of messages delivered. Among the three prediction-based methods, PRoPHET transmitted fewer messages, but all three had comparable delivery ratios as seen in Figure 6.2.

Figure 6.4 shows that epidemic and all prediction-based methods had substantial meta-data transmissions, though epidemic routing had relatively more (at least for shorter TTLs). Because the epidemic protocol transmitted messages at every contact, in turn, more nodes had messages that required meta-data transmission during contact. The direct-delivery and random protocols had no meta-data transmissions.

In addition to its message transmissions and meta-data transmissions, the epidemic routing protocol also had excessive message duplications, spreading replicas of messages over the network. Figure 6.5 shows that epidemic routing had one or two orders more duplication than the prediction-based protocols. Recall that the direct-delivery and random protocols did not replicate, thus had no data duplications.

Figure 6.6 shows the median delivery delays, and Figure 6.7 shows the mean delivery delays. All protocols show similar delivery delays in both mean and median measures for medium TTLs, but differ for long and short TTLs. With a 100-hour TTL, or unlimited TTL, epidemic routing had the shortest delays. Direct delivery had the longest delay for unlimited TTL, but it had the shortest delay for the one-hour TTL.

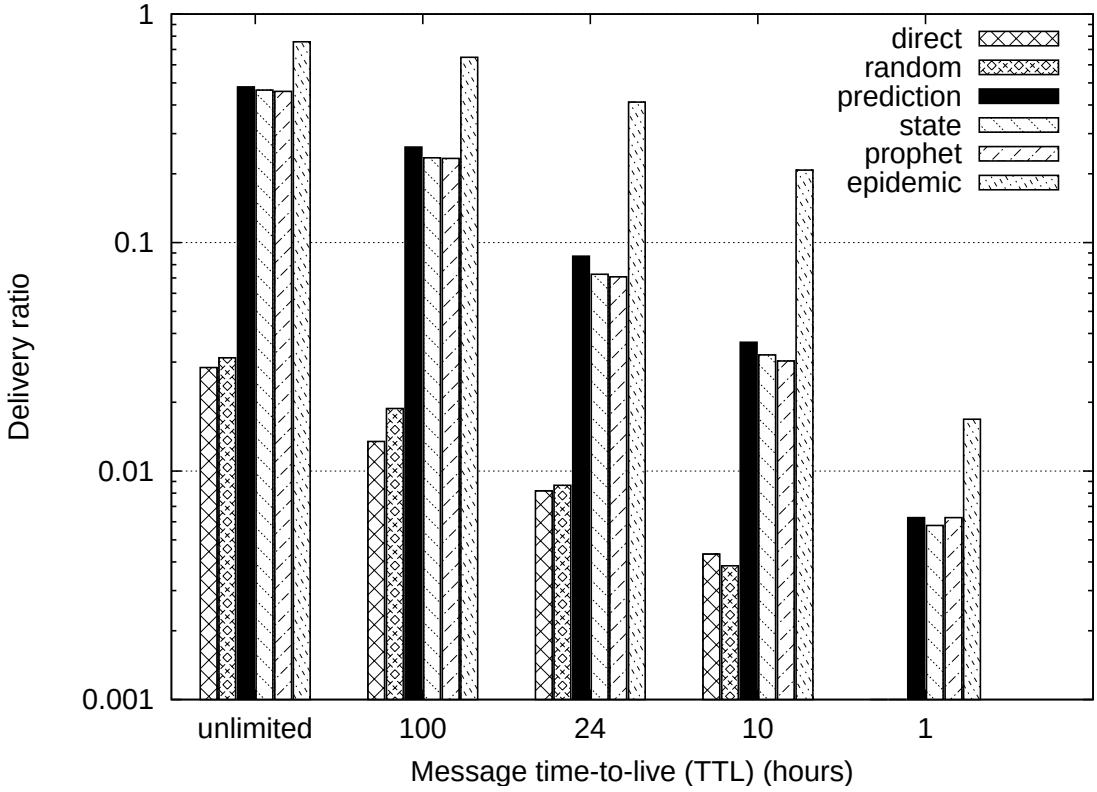


Figure 6.2: Delivery ratio (log scale). The direct and random protocols for one-hour TTL had delivery ratios that were too low to be shown in the plot.

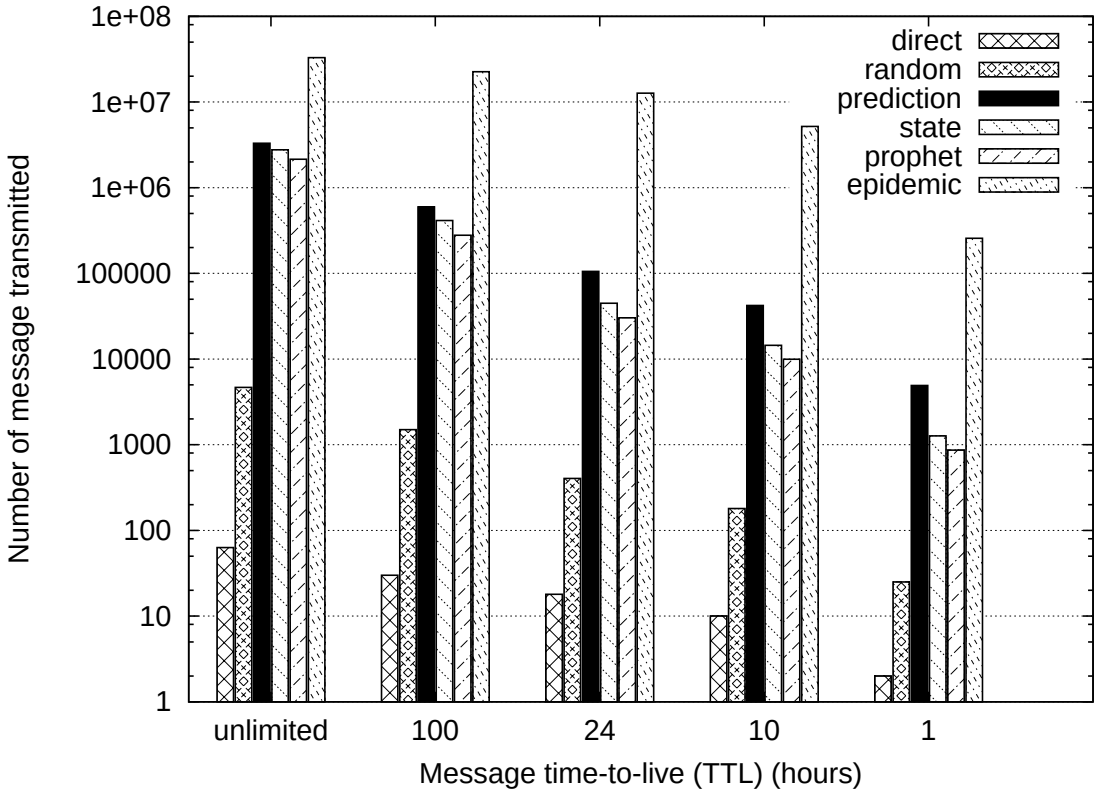


Figure 6.3: Message transmissions (log scale)

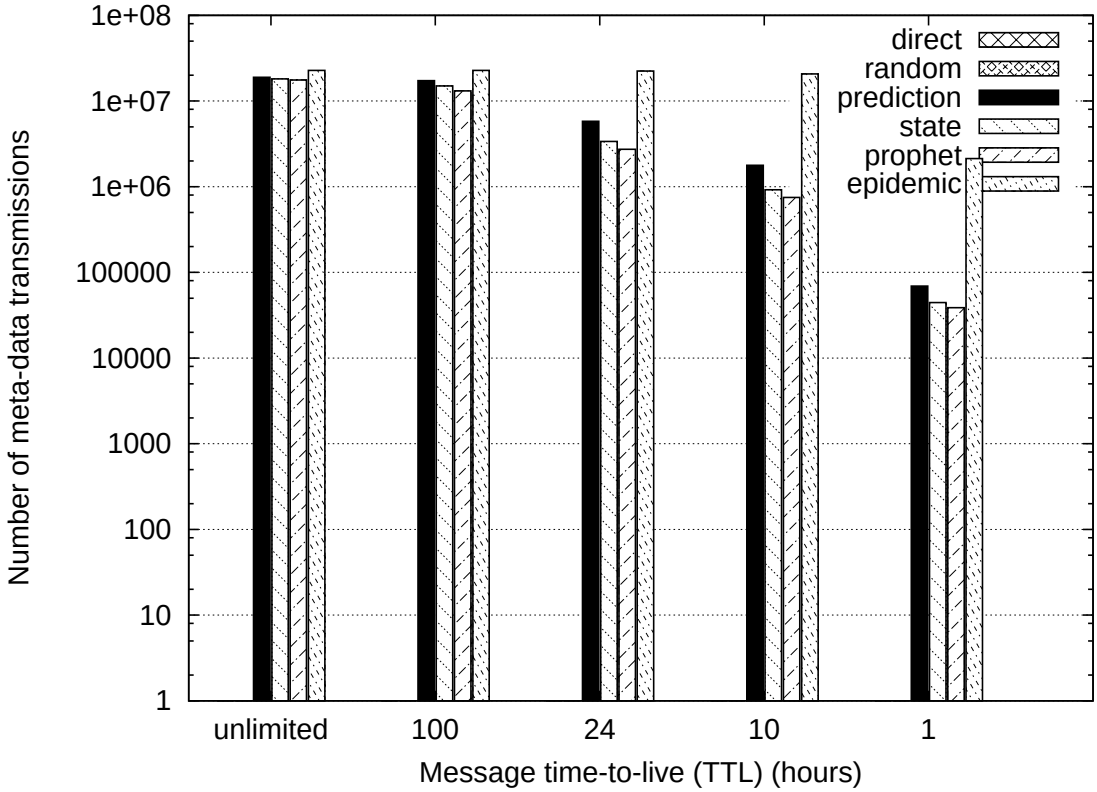


Figure 6.4: Meta-data transmissions (log scale). Direct and random protocols had no meta-data transmissions.

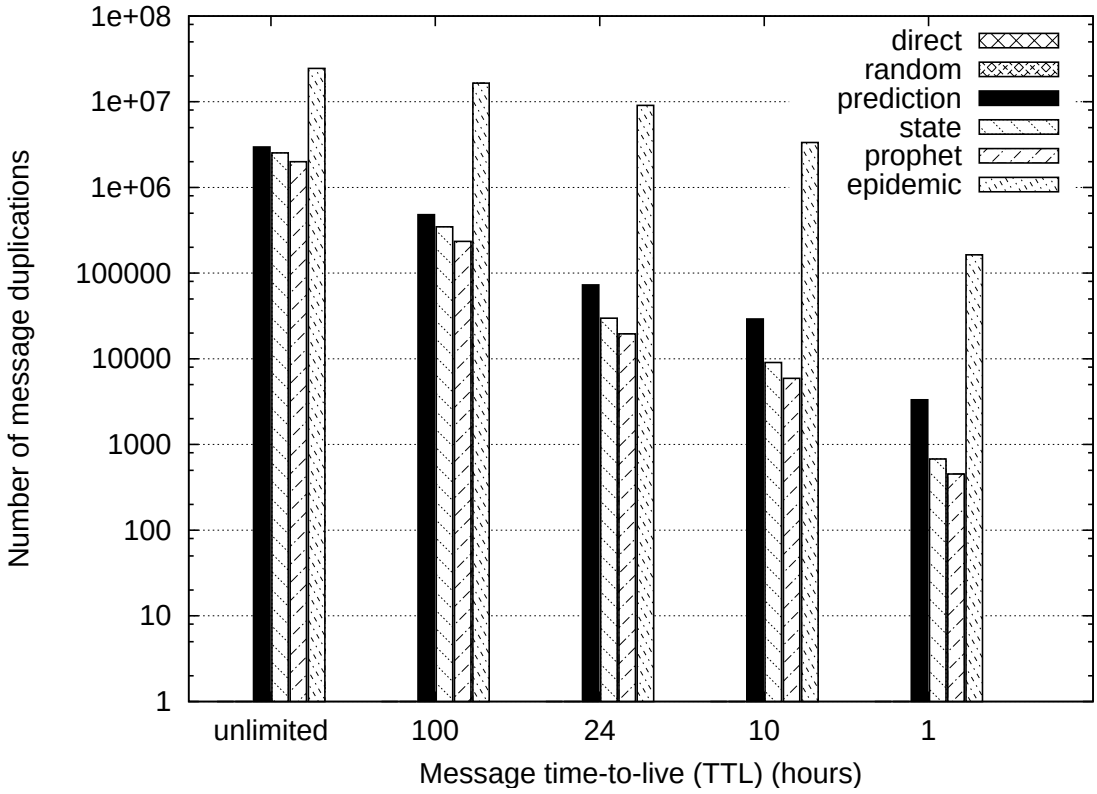


Figure 6.5: Message duplications (log scale). Direct and random protocols had no message duplications.

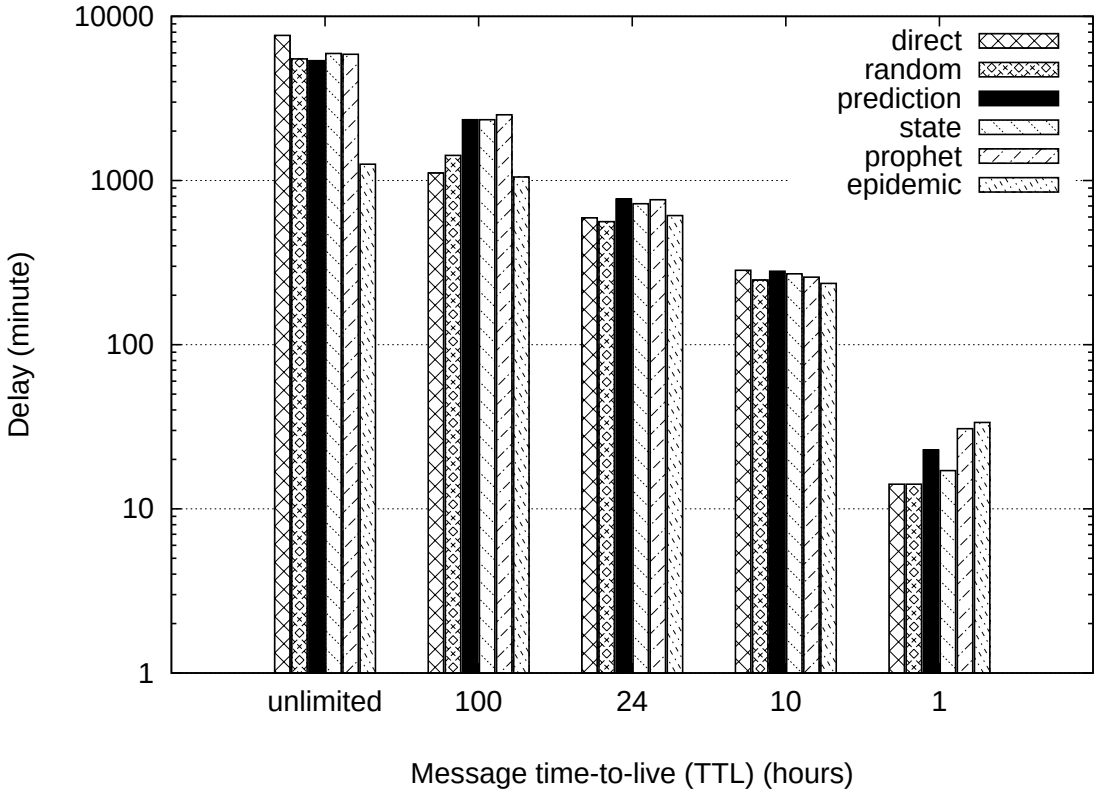


Figure 6.6: Median delivery delay (log scale)

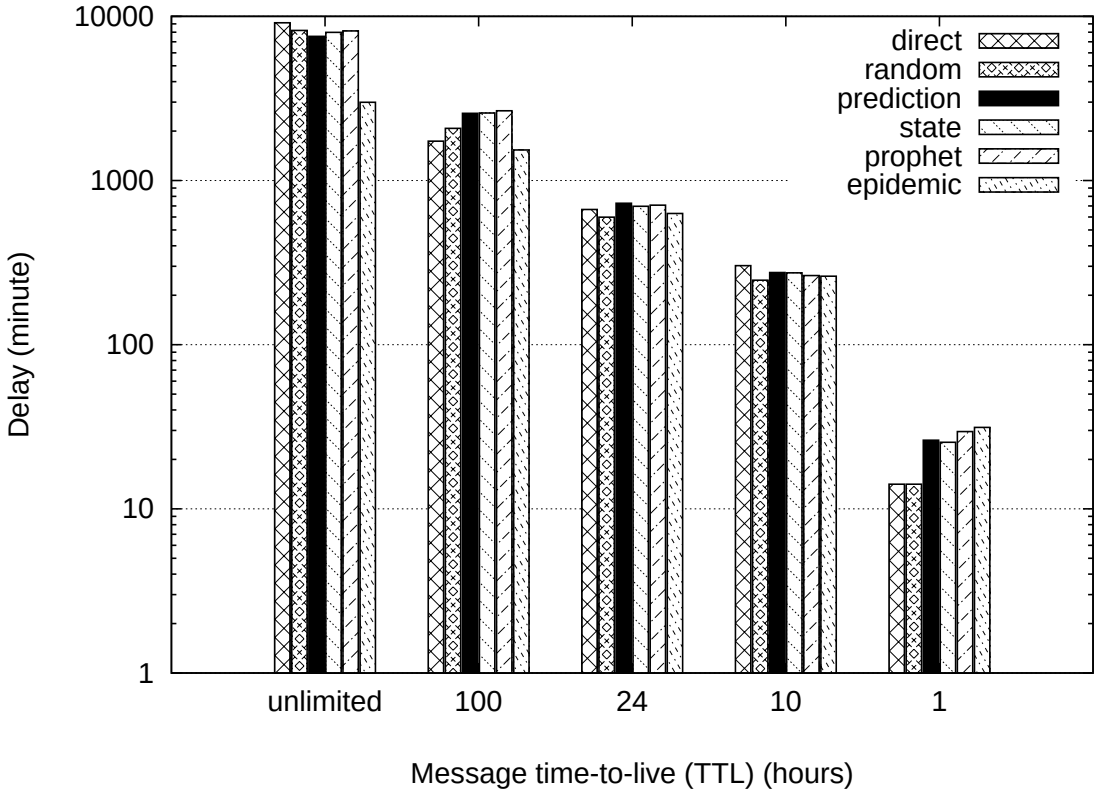


Figure 6.7: Mean delivery delay (log scale)

The results seem contrary to our intuition: the epidemic routing protocol should be the fastest routing protocol since it spreads messages all over the network. Indeed, the figures show only the delay time for *delivered* messages. For direct delivery, random, and the probability-based routing protocols, relatively few messages were delivered for short TTLs, so many messages expired before they could reach their destination; those messages had infinite delivery delay and were not included in the median or mean measurements. For longer TTLs, more messages were delivered even for the direct-delivery protocol. The statistics of longer TTLs for comparison are more meaningful than those of short TTLs.

Since our message generation rate was low, the storage usage was also low in our simulation. Figure 6.8 and Figure 6.9 show the maximum and average of maximum volume (in kbytes) of messages stored in each node. The epidemic routing had the most storage usage. The message time-to-live parameter was the big factor affecting the storage usage for epidemic and prediction-based routing protocols.

We studied the impact of different parameters of our prediction-based routing protocol. Our prediction-based protocol was sensitive to several parameters, such as the probability threshold. Figure 6.10 shows the delivery ratios when we used different probability thresholds. (The leftmost value 0.01 is the value used for the other plots.) A higher probability threshold limited the transfer probability, so fewer messages were delivered. We also had fewer transmissions, as shown in Figure 6.11. With a larger prediction window ΔT, we got higher contact probability. Thus, for a same probability threshold, we had higher a delivery ratio (Figure 6.12), and more transmissions (Figure 6.13). Our protocol was not so sensitive to the prediction window as it was to probability threshold. When the prediction window increased to one hour or longer, the delivery ratio and the number of message transmitted increased only slightly. Therefore, the contact probability within one hour or longer did not change much.

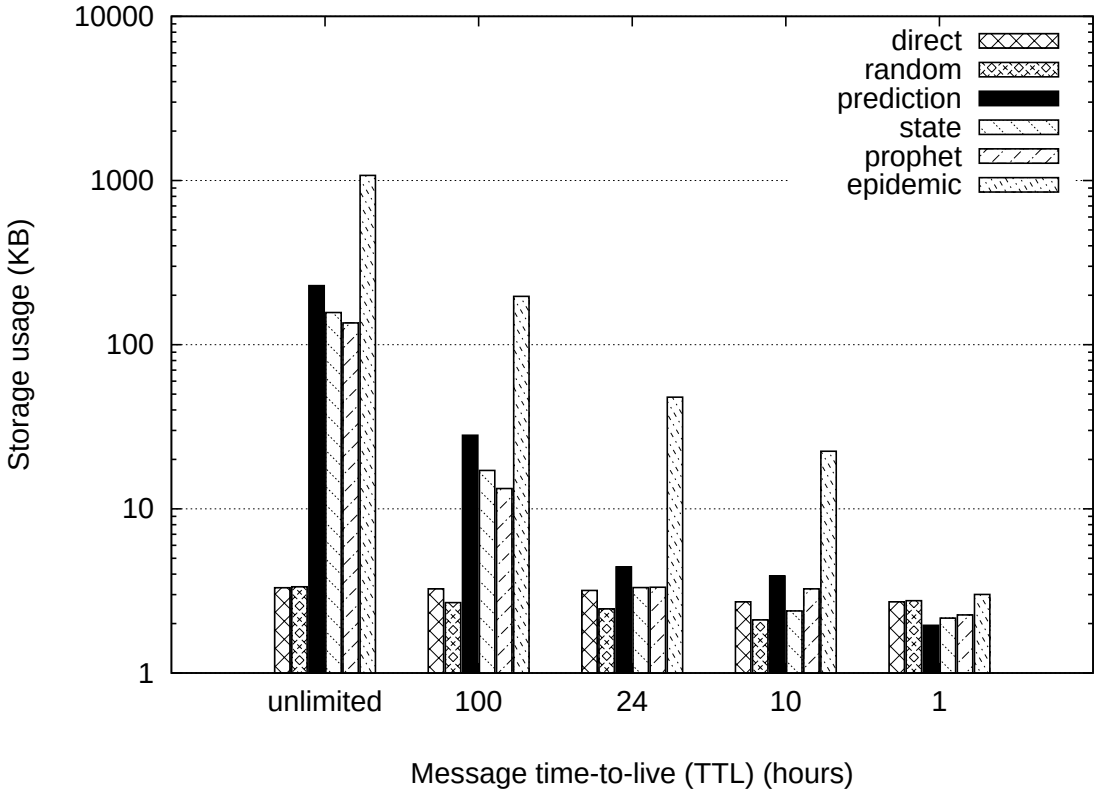


Figure 6.8: Max of maximum storage usage across all nodes (log scale).

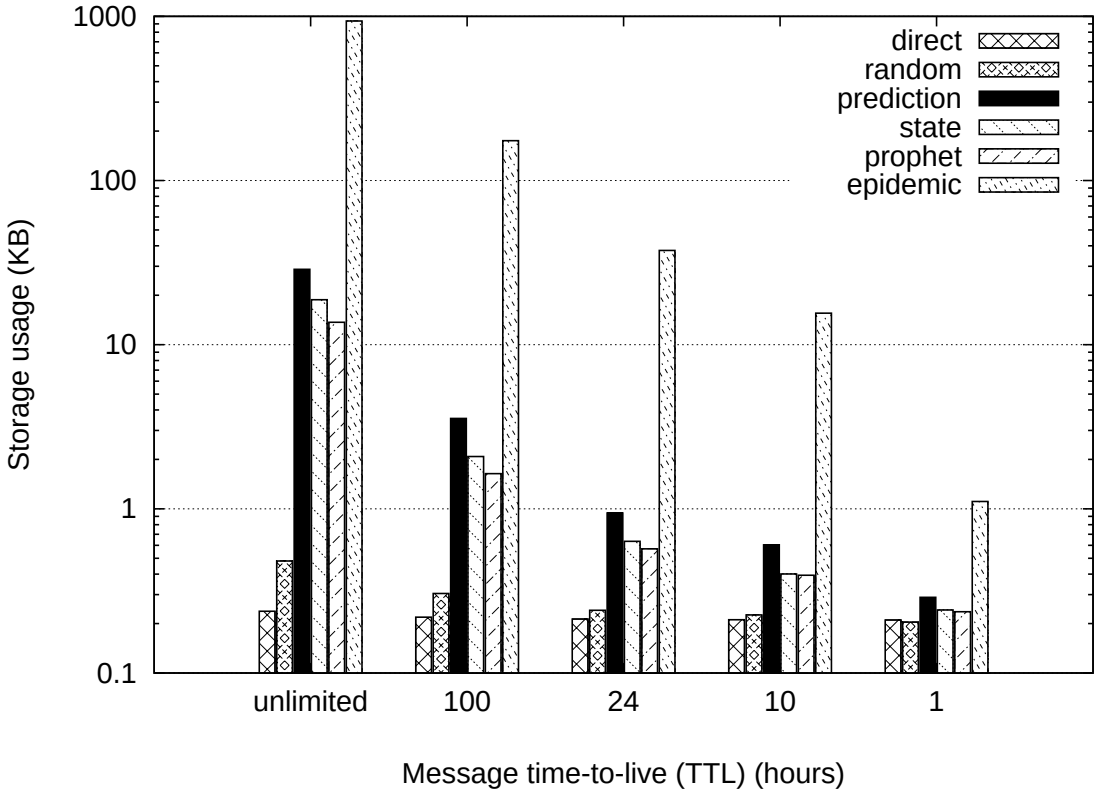


Figure 6.9: Mean of maximum storage usage across all nodes (log scale).

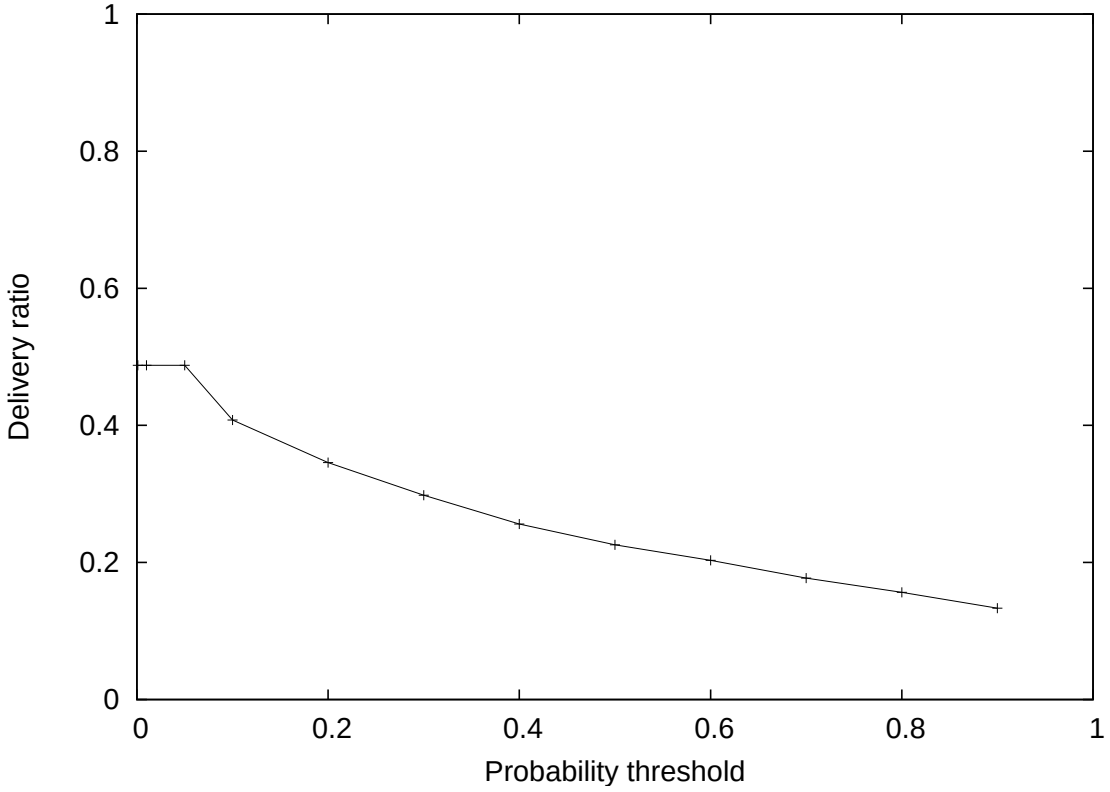


Figure 6.10: Probability threshold impact on delivery ratio of timely-contact routing. We used threshold 0.01 for other plots.

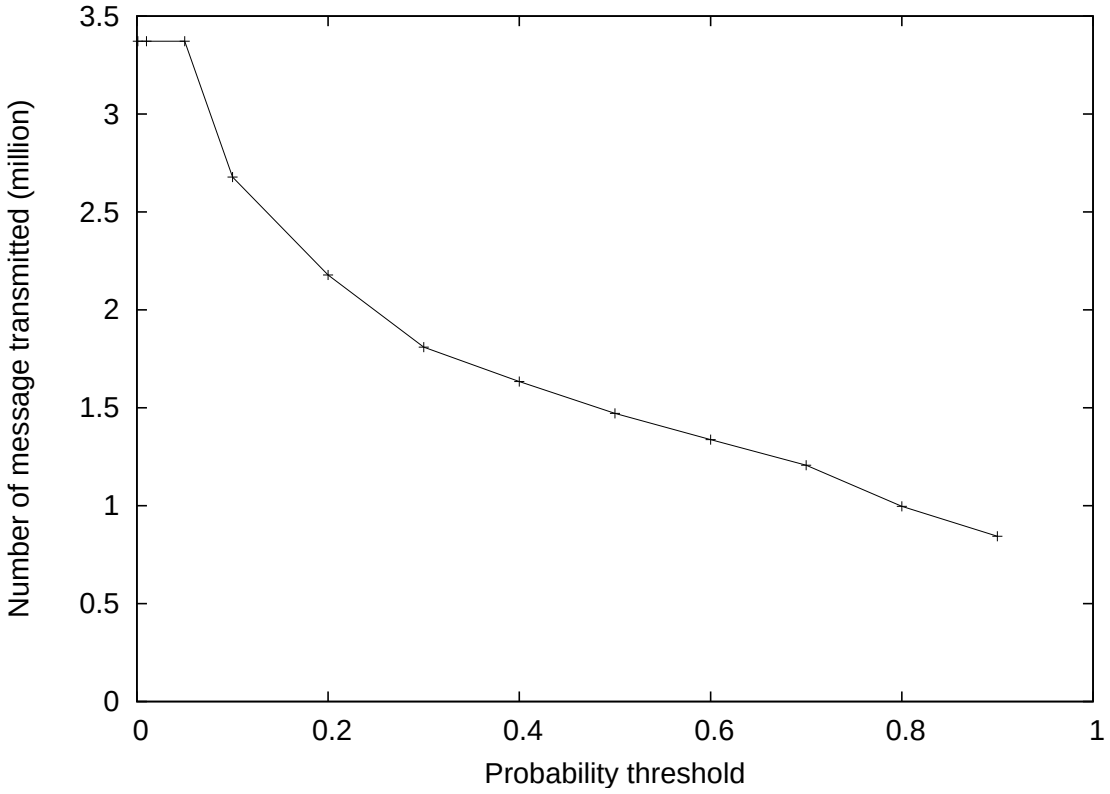


Figure 6.11: Probability threshold impact on message transmission of timely-contact routing. We used threshold 0.01 for other plots.

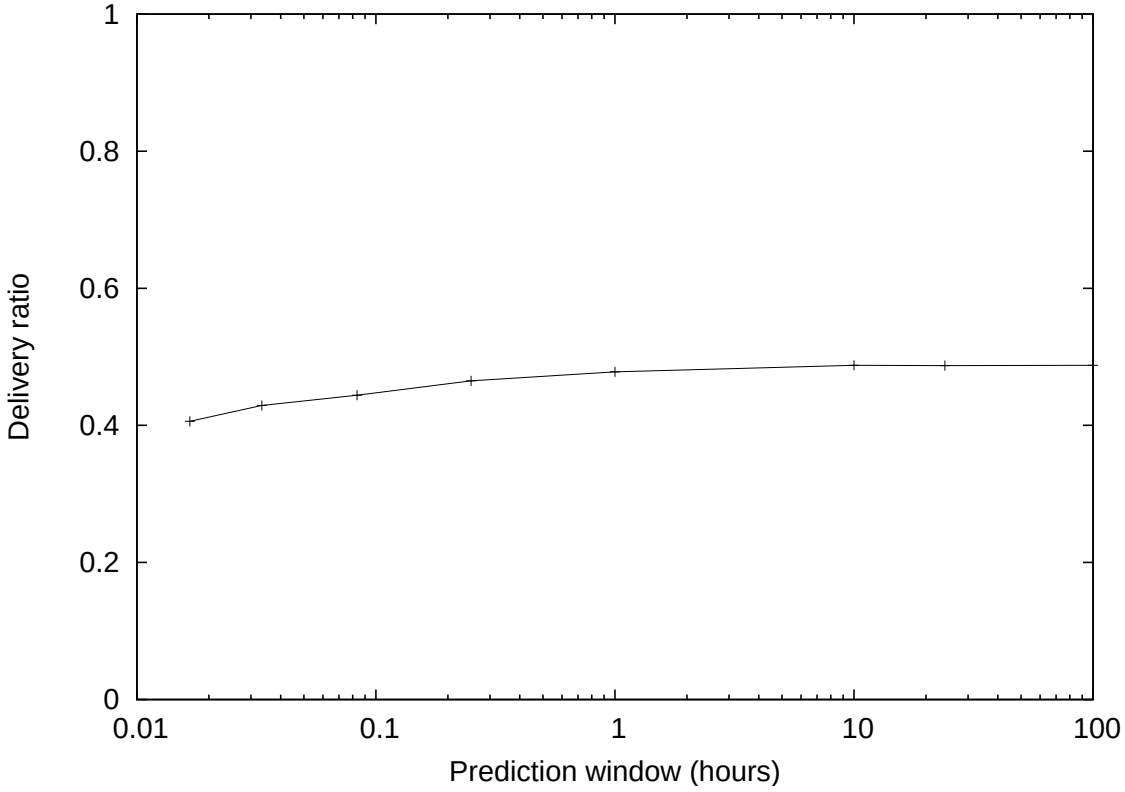


Figure 6.12: Prediction window impact on delivery ratio of timely-contact routing (semi-log scale). We used prediction window $\Delta T = 10$ hours for other plots.

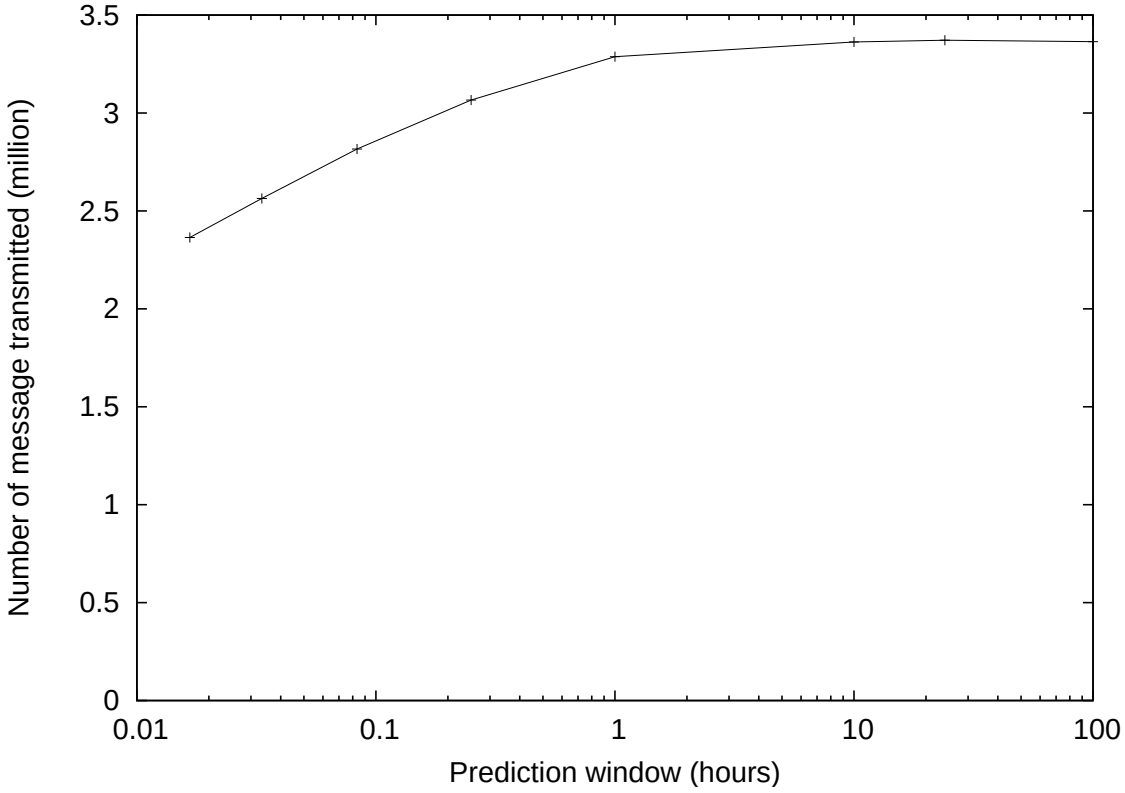


Figure 6.13: Prediction window impact on message transmission of timely-contact routing. We used prediction window $\Delta T = 10$ hours for other plots.

6.5 Related work

Fall [Fal03] presents an overview of DTNs. It gives examples of delay-tolerant networks: terrestrial mobile networks, exotic media networks (e.g., satellite, deep space), military ad-hoc networks, and sensor networks. The challenges of those networks are high latency, disconnection, and long queuing time. Fall focuses on the interoperability of heterogeneous networks (e.g., Internet, satellite networks, intranet, or ad-hoc networks). The author proposes to use a DTN gateway to connect different networks, and defines naming of network entities.

Chuah et al. [CCD05] extends the naming convention in Fall's DTN architecture framework [Fal03] to further divide a network region into groups. It also discusses details of neighbor discovery, gateway selection, mobility management, and route discovery.

Jain and Fall [JFP04] extend the DTN framework [Fal03]. They propose and evaluate several routing algorithms in two example scenarios: remote village and city bus. Both scenarios have predictable connectivity. The remote village has dial-up connection late at night, satellite connection every few hours, and a motorbike message courier every 4 hours. The city buses follow bus routes. Five routing algorithms are evaluated in the paper:

- First Contact (FC): a message is forwarded to one random current contact or to the first available contact;
- Minimum Expected Delay (MED): use Dijkstra's algorithm to compute the cost (delay) of each path and choose an edge that leads the path that has the least sum of the average waiting time, propagation delay and transmission delay;
- Earliest Delivery (ED): use modified Dijkstra with time varying cost, but without queue waiting time;
- Earliest Delivery with Local Queue (EDLQ): ED with the cost function incorporating

local queuing;

- Earliest Delivery with All Queue (EDAQ): ED with the cost function incorporating all nodes' queuing info;

- Linear Program (LP): use all the contacts, queuing, and traffic information.

They conclude that in networks with limited resources, the smarter algorithms (ED, EDLQ, and EDAQ) may provide a significant benefit. They also found that global knowledge may not be required for good performance, since EDLQ performed as well as EDAQ in many cases. The routing problem, however, when the contacts are predictable is relatively simple, especially in the remote village scenario, where the village communicates with only the city through three different routes. The LP algorithm provides only a theoretical analysis. In practice, the required information is not available.

Li and Rus [LR03] propose algorithms to guide mobile nodes' movements for communication in disconnected mobile ad hoc networks. Two algorithms were studied. The first assumes that mobilities and locations are known to all nodes. The second one is more generalized and does not assume this known information. Li and Rus describe a network situation in which nodes are moving with their tasks and may move to other nodes for message delivery. The proposed algorithms avoid traditional waiting and retry schemes. The big disadvantage is that the algorithms require users to move based on the needs of messages. This means that users (people) need to be aware of the messages and decide if relay is necessary. The algorithms are tools to aid the user to make the move decision.

Wang and Wu [WW06] studied two basic routing approaches: direct delivery and flooding. They propose a delivery probability based scheme. They found that their delivery-probability scheme achieves a higher delivery ratio than both of the basic routing approaches. They also studied the average delay and transmission overhead for all the studied delivery methods. Although the flooding approach has the lowest average delay, the deliv-

ery ratio suffers because excessive message duplications exhaust storage buffers. The authors discuss a mobile sensor network in which all sensor nodes transmit messages to sink nodes. Messages can be delivered to any sink node. The limited number of destinations enables the simple probability estimation of the proposed scheme.

LeBrun et al. [LCGZ05] propose a location-based, delay-tolerant network routing protocol. Their algorithm assumes that every node knows its own position, and the destination is stationary at a known location. A node forwards data to a neighbor only if the neighbor is closer to the destination than its own position. Our protocol does not require knowledge of the nodes' locations, and it learns their contact patterns.

Leguay et al. [LFC06] use a high-dimensional space to represent a mobility pattern, then routes messages to nodes that are closer to the destination node in the mobility pattern space. Node locations are required to construct mobility patterns.

Musolesi et al. [MHM05] propose an adaptive routing protocol for intermittently connected mobile ad hoc networks. They use a Kalman filter to compute the probability that a node delivers messages. This protocol assumes group mobility and cloud connectivity; that is, nodes move as a group, and among this group of nodes a contemporaneous end-to-end connection exists for every pair of nodes. When two nodes are in the same connected cloud, DSDV [PB94] routing is used.

Erasure-coding [JDPF05, WJMF05] explores coding algorithms to reduce message replicas. The source node replicates a message m times, then uses a coding scheme to encode them in one big message. After replicas are encoded, the source divides the big message into k blocks of the same size, and transmits a block to each of the first k encountered nodes. If m of the blocks are received at the destination, the message can be restored, where $m < k$. In a uniformly distributed mobility scenario, the delivery probability increases because the probability that the destination node meets m relays is greater than it meets k relays, given $m < k$.

We did not implement and evaluate these routing protocols, because either they require unrealistic future information [JFP04], controllable mobile nodes [LR03], domain-specific information (location information) [LCGZ05, LFC06], assume certain mobility patterns [MHM05], or present orthogonal approaches [JDPF05, WJMF05] to other routing protocols.

Chapter 7

Summary

In this chapter, we summarize our research on prediction techniques and their performance on several applications. We also discuss future directions of the research in this field. First, we summarize our conclusions.

We conducted the first comprehensive, empirical comparison of four important classes of location predictors (namely Markov, LZ, PPM, and SPM), to gauge which could most accurately predict the next cell for a mobile wireless-network user. The Markov predictors are easier to implement, use less memory, and run faster than the other predictors. We also found, surprisingly, that the complex predictors were only negligibly better than the simple Markov predictor, a result that has important implications for anyone developing or using such predictors in real applications.

In general, the simple low-order Markov predictors worked as well or better than the more complex compression-based predictors, and better than high-order Markov predictors. In particular, $O(3)$ (or above) Markov predictors did not improve over $O(2)$ Markov, and indeed reduced accuracy by failing to make predictions much of the time.

Most of the predictors, as defined in the literature, fail to make a prediction when faced with a context (recent history) that has never been encountered in the user's full history.

We found it was simple to add "fallback" to most predictors, allowing the predictor to use shorter and shorter context until it was able to make a prediction, and that this fallback often improved the predictor's accuracy substantially.

Next, we experimented with a variety of time predictors, which predict the time of a mobile user's next handoff event, and some jointly predict the destination as well. We select predictors such that their success would depend on certain dependencies across space and time. We further investigated whether group handoff or individual handoff models better represent the real situation. After evaluating these predictors with a suite of performance metrics, we found that no single predictor feature performed uniformly well, i.e., the prediction quality varies widely from user to user and from access point to access point. In the over- and under-provisioning results, for example, we found the distributions to be highly skewed.

Our results show that predicting the precise time of the handoff with a granularity in the order of seconds and minutes is in general difficult by using only the association history of the mobile devices. These results drive us to explore predictors that can return a range of values, can express confidence in their prediction (a probability), and can express inequalities (the probability that the visit will last at least t seconds, rather than predicting departure after precisely t seconds). We observe that predictions for individual events still tend to be highly skewed even for such *soft* prediction outputs. This motivates us to look into applications that could manage such skewness and benefit from soft values. Thus, we looked at a popular application, i.e., VoIP over WLAN, that can use per-user predictions to reserve resources for a group of users (in our case, the handoff users). Indeed, our results demonstrate that when coupled with intelligent predictors, VoIP performance improved significantly over the cases when only simple predictors are allowed or when no reservations are made at all.

Overall, our results show that we can predict some users more accurately while we

predict others less accurately. Even with only modest prediction accuracy, our simulation of voice telephone bandwidth reservation shows that we can improve the call drop-rate significantly without worsening the call block-rate much.

Finally, we simulated and evaluated several routing protocols in opportunistic networks.

We propose a prediction-based routing protocol for opportunistic networks. We evaluate the performance of our protocol using realistic contact traces, and compare to five existing routing protocols: three simple protocols and two other prediction-based routing protocols.

Our simulation results show that direct delivery had the lowest delivery ratio, the fewest data transmissions, and no meta-data transmission or data duplication. Direct delivery is suitable for devices that require extremely low power consumption. The random protocol increased the chance of delivery for messages otherwise stuck at some low-mobility nodes. Epidemic routing delivered the most messages. The excessive transmissions, and data duplication, however, consume more resources than portable devices may be able to provide.

Direct-delivery, random and epidemic routing protocols are not practical for real deployment of opportunistic networks, because they either had an extremely low delivery ratio or had an extremely high resource consumption. The prediction-based routing protocols had a delivery ratio more than 10 times better than that for direct-delivery and random routing, and they had fewer transmissions and used less storage than epidemic routing. They also had fewer data duplications than epidemic routing.

All the prediction-based routing protocols that we evaluated had similar performance. Our method had a slightly higher delivery ratio, but more transmissions and higher storage usage. There are many parameters for prediction-based routing protocols, however, and different parameters may produce different results. Indeed, there is an opportunity for some adaptation; for example, high-priority messages may be given higher transfer and

replication probabilities to increase the chance of delivery and reduce the delay, or a node with infrequent contact may choose to raise its transfer probability.

We must note that our evaluations were based solely on Dartmouth's traces[1]. Our findings and conclusions may or may not apply to other network environments, because users in different network environments may have distinct mobility patterns, and thus distinct handoff patterns. We also note that our traces were not motion records, our contact model were based on association records, and messages were synthetically generated. We believe, however, that our traces are a good match for the evaluation of handoff predictions and bandwidth reservations. The use of these traces for evaluating opportunistic network routing protocols may be a stretch, but not unreasonable. A more precise evaluation may need motion traces, that is, the physical location of users, and models to determine users' connectivity.

7.1 Contributions

This dissertation makes three major contributions. As our first contribution, we evaluate a series of handoff-location and handoff-time predictors that can take advantage of various dependencies across time and space. This approach not only provides a sanity check for widely used mobility and prediction models in the current state of the art, but also derives insight about the predictor features that drive good performance in a real WLAN environment. Note that our main concern here is not on the complexity of the predictors and their implementation details; rather our goal is to understand the level of predictability for the real handoff traces that we have. With this in mind, our study is an important report on whether a campus WLAN environment is amenable to accurate prediction of handoff events.

[1]Many of the traces we used are available from CRAWDAD [CRA].

As our second contribution, we present a case study that quantifies the possible gains due to handoff prediction in an actual application. In our case study, we focus on VoIP as the application of interest and we evaluate the use of handoff predictors for advance bandwidth reservation to maintain VoIP service quality after handoffs. We measure the performance using application-specific *call drop rate* and *call block rate* metrics. This approach allows us to dissect the performance gains due to the quality of predictions and the reservation policy. We focus on the influence on call drop rate and call block rate when handoff prediction applies to a variety of call admission schemes.

Our third contribution is that we propose a routing protocol for opportunistic networks and compared its performance to two other prediction-based routing protocols, as well as to three other basic routing protocols. We found that the three basic routing protocols in general were not practical for large opportunistic networks. The three prediction-based protocols can trade-off the delivery ratio against resource usage, and performed well and comparable to each other.

7.2 Future work

We have completed a comprehensive and empirical study that evaluated mobility predictors in wireless networks for bandwidth reservation and opportunistic routing. There are many opportunities for further research in this area.

- If we could obtain real calling records from a similar campus population, and map those calling patterns onto the handoff data set, we could reevaluate the bandwidth-reservation schemes with realistic calling patterns rather than the exponential distributions that we used in Chapter 5.

- We could seek ways to explicitly treat the "OFF" state in the handoff patterns, to

better predict exits from the network as distinct from normal cell-to-cell roams.

- Our interest has been limited to predicting inter-AP handoffs, but it is also of interest to predict location changes at a coarser level, e.g., building or sub-net.

- We could repeat our analysis on the predictability of user handoff in a campus network using location measurements that are collected at a finer granularity, e.g., using RFID, sensors, or signal-strength measurements.

- We could use traces other than the Dartmouth data to drive all three simulations. This effort may give more insight to the performance of prediction techniques and routing protocols.

- We could use domain-specific information, such as APs geographical location for handoff-prediction, or class registration information for DTN routing.

- We could theoretically analyze and compare the computational complexity of different prediction algorithms.

- We left out the technical details for implementing different bandwidth-reservation and call-admission systems. We could investigate whether centralized or distributed systems are suitable.

- We could study alternative messaging schemes in opportunistic networks, other than the unicast peer-to-peer messaging we assume here. For example, messages can be sent to a *location*, so that every node at that location will receive a copy of the message. Location prediction may be used to predict nodes' mobility, and to choose as relays those nodes moving toward the destined location.

- We could study the security and privacy issues in opportunistic networks. Many routing protocols require nodes to exchange historical mobility information for better

routing strategies. On one hand, users may be reluctant to expose their privacy, thus limiting the benefit of some routing strategies. On the other hand, malicious nodes may take advantage of some nodes, revealing too much information.

Bibliography

[AZ01] AbdulRahman Aljadhai and Taieb F. Znati. Predictive mobility support for QoS provisioning in mobile wireless environments. *IEEE Journal on Selected Areas in Communications (JSAC)*, 19(10):1915–1930, October 2001.

[BC03] Magdalena Balazinska and Paul Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In *Proceedings of the Annual International Conference on Mobile Systems, Applications, and Services (ACM Mobisys)*, pages 303–316, May 2003.

[BCW90] Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text Compression*. Prentice Hall, 1990.

[BD02] Amiya Bhattacharya and Sajal K. Das. LeZi-Update: An information-theoretic approach to track mobile users in PCS networks. *ACM/Kluwer Wireless Networks*, 8(2-3):121–135, March–May 2002.

[BGJL06] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. MaxProp: routing for vehicle-based disruption-tolerant networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, April 2006.

[BHT+03] Scott Burleigh, Adrian Hooke, Leigh Torgerson, Kevin Fall, Vint Cerf, Bob Durst, Keith Scott, and Howard Weiss. Delay-tolerant networking: An approach to interplanetary Internet. *IEEE Communications Magazine*, 41(6):128–136, June 2003.

[BNH03] Fan Bai, Narayanan, and Ahmed Helmy. IMPORTANT: A framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, volume 2, pages 825–835, March 2003.

[BRSS04] D. Bhattacharjee, A. Rao, C. Shah, and M. Shah. Empirical modeling of campus-wide pedestrian mobility: Observation on the USC campus. In *IEEE Vehicular Technology Conference (VTC)*, volume 4, pages 2887–2891, September 2004.

[BZB+97] R. Braden, L. Zhang, S. Berson, S. Herzon, and S. Jamin. Resource ReSerVation Protocol (RSVP) — version 1: Functional specification, September 1997. IETF RFC 2205.

[CBD02] Tracy Camp, Jeff Boleng, and Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[CCD05] Mooi Choo Chuah, Liang Cheng, and Brian D. Davision. Enhanced disruption and fault tolerant network architecture for bundle delivery (EDIFY). In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, 2005.

[CEL+06] Andrew Campbell, Shane Eisenman, Nicholas Lane, Emiliano Miluzzo, and Ronald Peterson. People-centric urban sensing. In *Proceedings of IEEE Wireless Internet Conference*, August 2006.

[CHC+06] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on the design of opportunistic forwarding algorithms. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, April 2006.

[CJv03] Christine Cheng, Ravi Jain, and Eric van den Berg. Location prediction algorithms for mobile wireless systems. In Borko Furht and Mohammad Illyas, editors, *Wireless Internet Handbook: Technologies, Standards, and Applications*, pages 245–263. CRC Press, 2003.

[CLP04] Francisco Chinchilla, Mark Lindsey, and Maria Papadopouli. Analysis of wireless information locality and association patterns in a campus. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, volume 2, pages 906–917, March 2004.

[CRA] CRAWDAD. Community Resource for Archiving Wireless Data at Dartmouth. http://crawdad.cs.dartmouth.edu.

[CS98] Sunghyun Choi and Kang G. Shin. Predictive and adaptive bandwidth reservation for hand-offs in QoS-sensitive cellular networks. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM)*, pages 155–166, August 1998.

[CT97] John G. Cleary and W. J. Teahan. Unbounded length contexts for PPM. *The Computer Journal*, 40(2/3):67–75, 1997.

[CW84] John G. Cleary and Ian H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, April 1984.

[DS99] Sajal K. Das and Sanjoy K. Sen. Adaptive location prediction strategies based on a hierarchical network model in a cellular mobile environment. *The Computer Journal*, 42(6):473–486, 1999.

[Fal03] Kevin Fall. A delay-tolerant network architecture for challenged Internets. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM)*, pages 27–34, August 2003.

[GKDJ05] Youngjune Gwon, James Kempf, Raghu Dendukuri, and Ravi Jain. Experimental results on IP-layer enhancement to capacity of VoIPv6 over IEEE 802.11b wireless LAN. In *Proceedings of the 1st Workshop on Wireless Network Measurements (WinMee 2005)*, April 2005.

[GPS+00] A. Ghanwani, W. Pace, V. Srinivasan, A. Smith, and M. Seamen. A framework for integrated sevices over switched and shared 802 LAN technologies, May 2000. IETF RFC 2816.

[HA06] Khaled A. Harras and Kevin C. Almeroth. Inter-regional messenger scheduling in delay tolerant mobile networks. In *Proceedings of IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, pages 93–102, June 2006.

[HCS+05] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe Diot. Pocket switched networks and human mobility in conference

environments. In *Proceedings of ACM SIGCOMM Workshop on Delay Tolerant Networking and Related Networks (WDTN)*, pages 244–251, August 2005.

[HKA04] Tristan Henderson, David Kotz, and Ilya Abyzov. The changing usage of a mature campus-wide wireless network. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (ACM Mobicom)*, pages 187–201. ACM Press, September 2004.

[JBRAS03] Amit Jardosh, Elizabeth M. Belding-Royer, Kevin C. Almeroth, and Subhash Suri. Towards realistic mobility models for mobile ad hoc networks. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (ACM Mobicom)*, pages 217–229. ACM Press, 2003.

[JDPF05] Sushant Jain, Mike Demmer, Rabin Patra, and Kevin Fall. Using redundancy to cope with failures in a delay tolerant network. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM)*, pages 109–120, August 2005.

[JFP04] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a delay tolerent network. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM)*, pages 145–158, August 2004.

[JLB05] Ravi Jain, Dan Lelescu, and Mahadevan Balakrishnan. Model T: an empirical model for user registration patterns in a campus wireless LAN. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (ACM Mobicom)*, pages 170–184, 2005.

[JM96] David B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, chapter 5, pages 153–181. Kluwer Academic, 1996.

[JOW+02] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li-Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 96–107, October 2002.

[JSA00] Philippe Jacquet, Wojciech Szpankowski, and Izydor Apostol. A universal predictor based on pattern matching, preliminary results. In Daniele Gardy and Abdelkader Mokkadem, editors, *Mathematics and Computer Science: Algorithms, Trees, Combinatorics and Probabilities*, chapter 7, pages 75–85. Birkhauser, 2000.

[KCC05] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. MANET simulation studies: The Incredibles. *Mobile Computing and Communications Review*, 9(4):50–61, 2005.

[KE05] David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. *Wireless Networks*, 11:115–133, 2005.

[KHA04] David Kotz, Tristan Henderson, and Ilya Abyzov. CRAWDAD data set dartmouth/campus. http://crawdad.cs.dartmouth.edu/dartmouth/campus, December 2004.

[KK05a] Minkyong Kim and David Kotz. Classifying the mobility of users and the popularity of access points. In *Proceedings of the International Workshop on*

Location- and Context-Awareness (LoCA), volume 3479 of *Lecture Notes in Computer Science*, pages 198–209. Springer-Verlag, May 2005.

[KK05b] Minkyong Kim and David Kotz. Modeling users' mobility among WiFi access points. In *Proceedings of the International Workshop on Wireless Traffic Measurements and Modeling (WiTMeMo '05)*, pages 19–24, June 2005.

[KL03] Hassan A. Karimi and Xiong Liu. A predictive location model for location-based services. In *Proceedings of the International Symposium of on Advances in Geographic Information Systems GIS*, pages 126–133, November 2003.

[KS03] Hyong S. Kim and Wee-Seng Soh. Mobility prediction for QoS provisioning. In *Mobile and Wireless Internet: Protocols, Algorithms and Systems*, chapter 4. Kluwer Academic Publishers, August 2003.

[KT04] Ioannis Z. Koukoutsidis and Michael E. Theologou. A combination of optimal partitioning and location prediction to assist paging in mobile cellular networks. *International Journal of Wireless Information Networks*, 11(3):123–129, July 2004.

[LAN95] David A. Levine, Ian F. Akyildiz, and Mahmoud Naghshineh. The shadow cluster concept for resource allocation and call admission in ATM-based wireless networks. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (ACM Mobicom)*, pages 142–150, November 1995.

[LAN97] David A. Levine, Ian F. Akyildiz, and Mahmoud Naghshineh. A resource estimation and call admission algorithm for wireless multimedia networks using the shadow cluster concept. *IEEE/ACM Transactions on Networking (TON)*, 5(1):1–12, February 1997.

[LB96] Songwu Lu and Vaduvur Bharghavan. Adaptive resource management algorithms for indoor mobile computing environments. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM)*, pages 231–242, August 1996.

[LBC98] Tong Liu, Paramvir Bahl, and Imrich Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks. *IEEE Journal on Selected Areas in Communication*, 16(6):922–936, August 1998.

[LCGZ05] Jason LeBrun, Chen-Nee Chuah, Dipak Ghosal, and Michael Zhang. Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks. In *Proceedings of IEEE Vehicular Technology Conference (VTC)*, pages 2289–2293, May 2005.

[LDS04] Anders Lindgren, Avri Doria, and Olov Schelen. Probabilistic routing in intermittently connected networks. In *Workshop on Service Assurance with Partial and Intermittent Resources (SAPIR)*, pages 239–254, 2004.

[LFC06] Jeremie Leguay, Timur Friedman, and Vania Conan. Evaluating mobility pattern space routing for DTNs. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, April 2006.

[LM96] George Liu and Gerald Maguire. A class of mobile motion prediction algorithms for wireless mobile computing and communications. *ACM/Baltzer Mobile Networks and Applications (MONET)*, 1(2):113–121, 1996.

[LR03] Qun Li and Daniela Rus. Communication in disconnected ad-hoc networks using message relay. *Journal of Parallel and Distributed Computing*, 63(1):75–86, January 2003.

[MHM05] Mirco Musolesi, Stephen Hailes, and Cecilia Mascolo. Adaptive routing for intermittently connected mobile ad hoc networks. In *Proceedings of IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, pages 183–189, June 2005. Extended version.

[MSA04] Arunesh Mishra, Minho Shin, and William A. Arbaugh. Context caching using neighbor graphs for fast handoffs in a wireless network. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, volume 1, pages 351–361, March 2004.

[MV05] Marvin McNett and Geoffrey M. Voelker. Access and mobility of wireless PDA users. *ACM Mobile Computing and Communications Review*, 9(2):40–55, April 2005.

[OLP] One Laptop Per Child (OLPC) project. http://laptop.org.

[PB94] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *Computer Communication Review*, pages 234–244, October 1994.

[PC04] S. Pack and Y. Choi. Fast handoff scheme based on mobility prediction in public wireless LAN systems. *IEE Proceedings - Communications*, 151(5):489–495, October 2004.

[PHA04] Prihandoko, M. H. Habaebi, and B. M. Ali. Aggregate history of user mobility pattern for QoS provisioning in multimedia wireless networks. *International Journal of Wireless Information Networks*, 11(1):19–27(9), January 2004.

[PR99] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.

[PSS05] Maria Papadopouli, Haipeng Shen, and Manolis Spanakis. Characterizing the duration and association patterns of wireless access in a campus. In *11th European Wireless Conference*, April 2005.

[RMSM01] Elizabeth M. Royer, P. Michael Melliar-Smith, and Louise E. Moser. An analysis of the optimum node density for ad hoc mobile networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 857–861, 2001.

[SB04] David Schwab and Rick Bunt. Characterising the use of a campus wireless network. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pages 862–870, March 2004.

[SC02] Sai Shankar and Sunghyun Choi. QoS signaling for parameterized traffic in IEEE 802.11e wireless LANs. In *Proceedings of Advanced Internet Services and Applications: First International Workshop, AISA*, pages 1683–1691, August 2002.

[SDK+06] Libo Song, Udayan Deshpande, Ulaş C. Kozat, David Kotz, and Ravi Jain. Predictability of WLAN mobility and its effects on bandwidth provisioning. In *Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Barcelona, Spain, April 2006. IEEE Computer Society Press.

[SG98] William Su and Mario Gerla. Bandwidth allocation strategies for wireless ATM networks using predictive reservation. In *Proceedings of IEEE*

Global Telecommunications Conference (GLOBECOM), volume 4, pages 2245–2250, November 1998.

[SGdL06] Jing Su, Ashvin Goel, and Eyal de Lara. An empirical evaluation of the student-net delay tolerant network. In *Proceedings of the International Conference on Mobile and Ubiquitous Systems (MobiQuitous)*, July 2006.

[SK01] Wee-Seng Soh and Hyong S. Kim. Dynamic guard bandwidth scheme for wireless broadband networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pages 572–581, April 2001.

[SK03] Wee-Seng Soh and Hyong S. Kim. QoS provisioning in cellular networks based on mobility prediction techniques. *IEEE Communications Magazine*, pages 86–92, January 2003.

[SK04] Wee-Seng Soh and Hyong S. Kim. Dynamic bandwidth reservation in cellular networks using road topology based mobility prediction. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pages 2766–2777, March 2004.

[SK05] Nancy Samaan and Ahmed Karmouch. A mobility prediction architecture based on contextual knowledge and spacial conceptual maps. *IEEE Transactions on Mobile Computing*, 4(6):537–551, 2005.

[SK07] Libo Song and David Kotz. Evaluating opportunistic routing protocols with large realistic contact traces. In *ACM MobiCom workshop on Challenged Networks (CHANTS 2007)*. ACM Press, September 2007.

[SKJH04] Libo Song, David Kotz, Ravi Jain, and Xiaoning He. Evaluating location predictors with extensive Wi-Fi mobility data. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 1414–1424, March 2004.

[SKJH06] Libo Song, David Kotz, Ravi Jain, and Xiaoning He. Evaluating next-cell predictors with extensive Wi-Fi mobility data. *IEEE Transactions on Mobile Computing*, 5(12):1633–1649, December 2006.

[SMY00] Xuemin Shen, Jon W. Mark, and Jun Ye. User mobility profile prediction: an adaptive fuzzy inference approach. *Wireless Networks*, 6(5):363–374, 2000.

[SN02] Samarth H. Shah and Klara Nahrstedt. Predictive location-based QoS routing in mobile ad hoc networks. In *Proceedings of IEEE International Conference on Communications (ICC)*, pages 1022–1027, April 2002.

[TB00] Diane Tang and Mary Baker. Analysis of a local-area wireless network. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (ACM Mobicom)*, pages 1–10. ACM Press, 2000.

[TB02] Diane Tang and Mary Baker. Analysis of a metropolitan-area wireless network. *Wireless Networks*, 8(2/3):107–120, 2002.

[VB00] Amin Vahdat and David Becker. Epidemic routing for partially-connected ad hoc networks. Technical Report CS-2000-06, Duke University, July 2000.

[VK96] Jeffrey Scott Vitter and P. Krishnan. Optimal prefetching via data compression. *Journal of the ACM*, 43(5):771–793, 1996.

[WJMF05] Yong Wang, Sushant Jain, Margaret Martonosi, and Kevin Fall. Erasure-coding based routing for opportunistic networks. In *Proceedings of ACM*

SIGCOMM Workshop on Delay Tolerant Networking and Related Networks (WDTN), pages 229–236, August 2005.

[WW06] Yu Wang and Hongyi Wu. DFT-MSN: the delay fault tolerant mobile sensor network for pervasive information gathering. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, April 2006.

[YL02] Fei Yu and Victor C. M. Leung. Mobility-based predictive call admission control and bandwidth reservation in wireless cellular networks. *Computer Networks*, 38(5):577–589, 2002.

[ZL78] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, September 1978.